THE DATA BASE MANAGEMENT SYSTEM

USER MANUAL AND EXAMPLE

by

Michel J. Bastarache
Ernest Allen Hershey III

ISDOS Working Paper No. 89
Preliminary Draft
April 1975

ISDOS Research Project
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48104
(313/763-3469)

THE UNIVERSITY OF MICHIGAN
ENGINEERING LIBRARY

This is a working paper and should not be quoted or reproduced in whole or in part without the written consent of the authors. Comments are solicited and should be addressed to the authors.
Preface

The purpose of this paper is to serve as an aid in using the DBMS subroutines specified by Ernest Allen Hershey in ISDOS Working Paper No. 88, "A Data Base Management System for PSA Based on DBTG 71."

It is assumed that the reader is familiar with the contents of this paper and that it is available for reference. A thorough understanding of FORTRAN is also assumed. This Working Paper specifies the use of Version D2.0 of the Data Base Management System (FORTRAN Implementation). Note that data bases generated by Version D1.0 (Assembler Implementation) are not compatible with Version D2.0.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>i</td>
</tr>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1. Overview of the System</td>
<td>4</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Implementation</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Hardware and Software Considerations</td>
<td>5</td>
</tr>
<tr>
<td>1.4 Logical and Physical Data Structures</td>
<td>5</td>
</tr>
<tr>
<td>1.5 System Functions</td>
<td>5</td>
</tr>
<tr>
<td>2. Specifying a Data Structure</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Grouping Data into Records</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Relating Records</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Contents of the Presidential File (PRESFILE)</td>
<td>11</td>
</tr>
<tr>
<td>2.4 A Data Structure for the Presidential Database</td>
<td>16</td>
</tr>
<tr>
<td>3. Specifying the DDL for a Data Structure</td>
<td>20</td>
</tr>
<tr>
<td>3.1 Naming Conventions</td>
<td>20</td>
</tr>
<tr>
<td>3.2 Records and Items</td>
<td>20</td>
</tr>
<tr>
<td>3.3 Item Size</td>
<td>25</td>
</tr>
<tr>
<td>3.4 Handling Relationships Among Records</td>
<td>25</td>
</tr>
<tr>
<td>3.5 Handling Groups</td>
<td>27</td>
</tr>
<tr>
<td>3.6 Handling Repeating Items</td>
<td>28</td>
</tr>
<tr>
<td>3.7 Handling Repeating Groups</td>
<td>28</td>
</tr>
<tr>
<td>3.8 Specifying Data Base Size</td>
<td>29</td>
</tr>
<tr>
<td>3.9 Developing the DDL for the Presidential Example</td>
<td>29</td>
</tr>
<tr>
<td>4. Creating and Initializing the Data Base</td>
<td>37</td>
</tr>
<tr>
<td>4.1 Creation of DDL, DBTF and DBF Files</td>
<td>37</td>
</tr>
</tbody>
</table>
Table of Contents (cont'd)

4.2 Generation of the DBTF .................................. 37
4.3 Initialization of the DBF ................................. 37

5. Coding Conventions for Data Base Access Programs ........ 38
   5.1 Common Areas and Block Data ......................... 38
   5.2 Buffer Variables ..................................... 41
   5.3 Access of the DBF and DBTF ............................ 41
   5.4 General Program Formats ............................... 42

6. Populating the Data Base ................................... 43
   6.1 Reading Data from the Input File ....................... 43
   6.2 Storing Data .......................................... 44
   6.3 FMSK .................................................. 44
   6.4 CR ...................................................... 44
   6.5 SFR ..................................................... 45
   6.6 AMS ..................................................... 45
   6.7 SRM ..................................................... 45
   6.8 SOM ..................................................... 46
   6.9 SMOVE .................................................. 46
   6.10 Code to Populate the President Data Base .......... 47

7. Generating Reports from the Data Base ..................... 70
   7.1 Retrieving Data ....................................... 70
   7.2 FFM ..................................................... 70
   7.3 FNM ..................................................... 70
Table of Contents (cont'd)

7.4 SOM ........................................... 71
7.5 SMM ........................................... 71
7.6 GFM ........................................... 71
7.7 GFO ........................................... 72
7.8 Report Routines .............................. 72
7.9 HEDING. ....................................... 72
7.10 BUFBLD ....................................... 72
7.11 NINBUF ...................................... 73
7.12 PBUF ......................................... 73
7.13 Code to Generate Reports from the Presidential Data Base ... 73

Appendix A  Presidential File (PRESFILE) Listing .................. 85
Appendix B  Presidential Report Examples ......................... 96
INTRODUCTION

Before any attempt is made to explain the implementation of this Data Base Management System (DBMS), definitions for the terms to be used must be presented. The following terms will be used throughout the paper.

- type (vs. occurrence)
- item
- group
- record
- set

DDL
DML

type (vs. occurrence)  When describing various objects in this paper it is necessary to differentiate between a "type" of object (i.e., its class) and an instance of one object within a "type" (i.e., an "occurrence"). For example, Social Security number may be a "type" of item, but an "occurrence" of a Social Security number might be 366-60-5321. This is a very important concept and comprehension of the difference is essential.

item
The "item" is the elementary data unit from which all other types of structures are ultimately composed. An occurrence of an "item" is the representation of a value which may be a number, a string of characters, a truth value, etc. Examples of types of items are age, hair, color, occupation, etc. Occurrences of these types of items might be: 35, BROWN and TEACHER, respectively. (An item is often called a field, data item or element in other DBMS's.)

group
A "group" is a named collection of "items" and/or other "groups." An occurrence of a group is an instance of a value for each item contained within it. Examples of types of possible groups are name (which might consist of first name, initial and surname), and date (consisting of month, day, year). Occurrences of these types of groups might be: JOHN Q. ADAMS and January 1, 1972, respectively. (A group is often called a segment or data aggregate.)

record
A "record" is a named collection of "items" and/or "groups." There may be an arbitrary number of occurrences in the data bases of each record type. The record is used to represent the major entities of an application. For example, if the system being constructed were a payroll system, it is very likely that there would be an occurrence of an EMPLOYEE-RECORD for each employee on the payroll.
set

A "set" is a named collection of "records" which specifies an ordering or relation among the record types within it. The specification of "owner records" and "member records" within the set designates the direction of the relationship. For example, if records were assigned to correspond to each person and each state in the U.S., a relationship between the two record types might be defined to associate all those people born in the same state. Since a person can only be born in one state and states can have several people born in them, the direction of the relationship is specified by designating state records as owner records and people records as member records (i.e., for any occurrence of a state record, there can be many occurrences of member records in the relationship.)

DDL

The "DDL" (Data Definition Language) is a language used to define data structures within a data base, usually based on terminology similar to that specified previously.

DML

The "DML" (Data Manipulation Language) is a language used to manipulate the data in the data base defined by the DDL. (For this particular system they are the subroutines described in Working Paper No. 88.)

The example that will be referenced throughout this paper is called the "Presidential Data Base." The input to our problem is the Presidential File (PRESFILE) which contains data about each President, from George Washington to Lyndon Johnson. Our goal is to implement a system which will store all this data into a data base in a form easy for retrieval. Finally, reports will be generated from the contents of the data base. The following types of data are contained in the PRESFILE and must be incorporated into the data base:

President data
Administration data
Congress data
Election data
State data

If one type of record could be defined to correspond to each of these data types, the data structure might look like Figure 1:

![Figure 1](image-url)
Any given President (occurrence of a President record) might be related to several Elections (occurrences of Election records) by the fact that the President won those elections. (By the previously defined terminology, the President records can be defined to be "owner records" for a "set" where the "member records" are Election records. This "set" could be called ELECTION-WON.) By the same token, a State record could be related to one or more President records by the fact that they were born there, and one or more State records could be related to Administration records by the fact that those states were admitted to the Union during that Administration, etc.

These relationships can be specified more clearly by a DDL format;

```
SET Elections-Won
  OWNER President
  MEMBER Election

SET Presidents-Born
  OWNER State
  MEMBER President

SET Administrations-Held
  OWNER President
  MEMBER Administration

SET States-Admitted
  OWNER Administration
  MEMBER State
```

Notice that all the relationships (sets) have been defined so that for any given occurrence of an owner record, there exists one or more occurrences of a member record (which can be notated as a 1:M relation). Utilizing this convention makes the task of specifying the DDL that much easier.

Each record type defined (President, Election, etc.) will consist of smaller collections or units of data. The Presidential record may contain information about date of birth, age, etc. Date would be defined as a "group" as it could be broken down into the "items": month, day and year. Depending on the facilities allowed in the DDL, specification of groups may or may not be allowed. The DDL presented in this system does not allow for groups, so all data within the Presidential record must be defined in terms of "items."
1. OVERVIEW OF THE SYSTEM

1.1 Background

This system was designed based on the specifications given by the CODASYL Data Base Task Group in their "CODASYL-COBOL Data Base Facility Proposal". It was developed and implemented by Ernest A. Hershey at the University of Michigan in the summer of 1973. Since then the system has been used in several independent applications:

Problem Statement Analyzer - The DBMS is used to maintain the data bases needed for the PSL/PSA System. (See ISDOS Working Paper No. 86 for more information about this application.) *

Computerized Bibliography - The DBMS is currently being used to maintain a data base consisting of an up-to-date bibliography for the ISDOS Project. *

Tape Library System - The DBMS is being used to maintain a tape library for another project at the University. This particular system has been set up on Michigan's 370/168 as well as on a Honeywell machine.

Presidential Data Base Example - To illustrate some of the capabilities of the DBMS this example has been implemented based on the data presented by A. Vorhaus and A. Weinert of the CODASYL Systems Committee. *

1.2 Implementation

This system is a host language Data Base Management System (DBMS). A system with host language capability is one which is built upon the facilities of a procedural language such as FORTRAN. The DBMS functions** are invoked from within the host language program written in COBOL or FORTRAN. The method of interface between system and program (be it either FORTRAN or COBOL) is via the CALL statements in the program. For this reason, the user must be somewhat experienced in programming.

The user of the system is considered to be a programmer in the sense that a set of FORTRAN (or COBOL) statements must be written to be executed sequentially. The user exercises the same degree of procedural control over processes as if programming in the host language except that the facilities of the DBMS handles data transfers to and from the data base.

This also means that all the capabilities of the host language are at the user's disposal. The high level DBMS functions such as UPDATE, DELETE, etc., must also be written by the user for a particular implementation of the system.

* This application is currently operative on the IBM 370/168 at the University of Michigan.

** In this sense "functions" means retrieval, storage, etc.
1.3 Hardware and Software Considerations

The DBMS described in this paper can be set up on any machine which has an ANSI FORTRAN or ANSI COBOL compiler. This makes the DBMS machine independent and operating-system independent. Mode of operation (on-line or batch) is of course dependent on the facilities of the installation.

1.4 Logical and Physical Data Structures

All types of logical data structures, from simple lists to complex network structures may be defined when using the DBMS. The Presidential database example presents an implementation of a network structure.

Any logical data structure defined by the user is translated into a doubly linked list structure for each "set" defined in the database. Records are stored according to a "best fit" algorithm and are identifiable by a unique "data base key". The number of records and sets that can be handled by the DBMS is, for all practical purposes, limited only by the amount of storage available.

1.5 System Functions

The method of defining the data base is shown in Figure 1:

1a. Interaction of the DBMS with the user in initializing the data base:

The user must specify the DDL for a particular usage of the system. This DDL is then to be used as input to a program which formulates a set of tables. These tables are used as input to another program which prepares the data base for accepting data. The tables are also used when referencing the data base for modification and retrieval purposes.
The method of populating the database or modifying it in any way is pictured in Figure 1b.

1b. Interaction of the DBMS with the user in modifying the contents of the database:

The user writes a program utilizing routines in the DBMS library. These routines also provide the means of interfacing the user's program with the Data Base Table File (DBTF), generated in the creation phase, and the database. Error messages are generated by the DBMS routines should errors occur in modifying the database.

The method of retrieving data from the database in the form of reports is shown in Figure 1c.

1c. Interaction of the DBMS with the user in generating reports from the database:
The user must write a program which calls routines in the DBMS library. These routines provide the means of referencing the DBTF for access to the data base and retrieving the data. The errors encountered while retrieving data will be documented on the user's outputs.
2. SPECIFYING A DATA STRUCTURE

Before the user writes the DDL for a particular system some preliminary considerations must be taken into account. These are: how is the data to be conceptually grouped (what will be defined to be records) and how will these groupings relate to each other. The user's conception of how this data is related (as opposed to how it is actually defined in the DDL or how it is physically stored) is called the "data structure".

2.1 Grouping Data into Records

The first decision to be made is how to group the data. This can be determined after studying the requirements for the system. In most cases grouping of data is trivial as all the data pertaining to one type of object (such as a person) would be grouped together. For example, if your system was concerned with maintaining information about people, it would be very likely that data used in identifying each person would be grouped together (i.e., name, social security number, etc.)

When it is not obvious whether to add data to one defined record or define a new one to contain it, the following guidelines may be helpful.

i. Attempt to reduce the degree of redundancy in the data structure (i.e., whenever possible store data in only one record rather than two).

ii. Define records (and the data within them) according to the requirements for retrieving this data. For example, if an inventory system were being developed and one of the output requirements was to generate a report consisting of all suppliers of items in the inventory, it would be a good idea to have a record type associated with suppliers (as opposed to storing the supplier information as part of the records for each item in the inventory).

After determining what data is to be contained in which records, a decision must be made as to how this data is grouped within the record, if at all. For example, it may be convenient to think of all data pertaining to a person's family as MARRIAGE-DATA which could be defined then as a "group" of items (i.e., marriage date, spouse's name, number of children, etc.).

If several occurrences of a particular group were possible for a single record occurrence (as in the case of MARRIAGE-DATA where a person could be married more than once) this type of data would be called a "repeating group". Likewise, a "repeating item" is a simple case of the repeating group where the group consists of only one item (e.g., OCCUPATION).
The end result of the data grouping process is a number of defined record types and specification of the data within them in terms of items, groups, repeating items and repeating groups for each record type.

For a simple inventory system the result could be shown by the following format.

<table>
<thead>
<tr>
<th>item record</th>
<th>item identifier</th>
<th>item name</th>
<th>color</th>
<th>weight</th>
<th>cost</th>
<th>date of last order</th>
<th>number in stock</th>
<th>location stored</th>
</tr>
</thead>
</table>

"item"

<table>
<thead>
<tr>
<th>supplier record</th>
<th>supplier identifier</th>
<th>supplier name</th>
<th>address</th>
<th>date of last order</th>
</tr>
</thead>
</table>

"group" "item" "repeating group"

<table>
<thead>
<tr>
<th>customer record</th>
<th>customer identifier</th>
<th>customer name</th>
<th>address</th>
<th>money owes</th>
<th>credit status</th>
</tr>
</thead>
</table>

Note that most data could be defined as "items". "Date of last order" was defined as a "group" since it consists of month, day and year and "location" is a "repeating group" because any given inventory item might be stored at more than one location and a given location might be identified by warehouse number, row number, rack number, etc.

2.2 Relating Records

The next step in specifying the data structure is to define the relationships among the records defined.

Between any two records that are to be related the type of relationship should be determined as 1:1, 1:M or M:M.
1:1 means that for every occurrence of a particular record type there is one and only one occurrence of another record type related to it. For example, all currently married men and women in the United States could be paired by the relationship of "marriage". Since any of these people are allowed only one spouse, the relationship would be designated as a 1:1 relationship.

1:M means that for every occurrence of a particular record type there can be zero, one or many occurrences of another record type related to it. In the inventory example, this type of relationship exists between the supplier and item records. A particular supplier could supply one or more of the items in the inventory.

M:M means that for an occurrence of a particular record type there can be many occurrences of a second record type related to it and likewise any occurrence of the second record type could be related to several occurrences of the first record type. This type of relationship is apparent when relating the item records to the customer records. Several items can be bought by one customer and several customers could buy the same type of item.

The relationships for the inventory example can be shown in the following format.

```
SUPPLIER RECORDS

supplies (1:M)

ITEM RECORDS

bought by (M:M)

CUSTOMER RECORDS
```

The titles in lower case present a name for the relationship and the arrow specifies the direction. The "type" of relation is contained in parentheses. This diagram then says that a particular supplier "supplies" one or more items in the inventory which are "bought by" many customers.
A relation which specifies an order among the record occurrences of one or more record types is called an ordering relationship. Specifying this type of relationship aids in retrieving particular record occurrences and in formatting any output reports (the data is already ordered). Ordering relationships for the inventory data structure can be shown in the following way.

supplier
order

SUPPLIER RECORDS

item
order

ITEM RECORDS

customer
order

CUSTOMER RECORDS

In most cases a particular "item" in the record will be used to designate order (the identifier "item" usually, but order could just as well be established on weight or cost for inventory items records).

2.3 Contents of the Presidential File (PRESFILE)

Before attempting to specify a data structure for the Presidential Data Base an inspection of PRESFILE must be made to see what data is available. The complete listing of all the data in PRESFILE is given in Appendix A. In summary, PRESFILE presents five different types of data groupings which are identified by the names:

PRES
ELECTION
ADMIN
CONGRESS
STATES

Following each identifier is a collection of data in a specific format, representing one occurrence of a data type. PRES, for example, designates that all following data (up to the next identifier) presents information about a particular president.
ELECTION designates that the following data pertains to a particular election; ADMIN, to a particular administration; CONGRESS to a particular congress; and STATES to a particular state.

The format of the data * following a PRES identifier is given below with respect to an actual example:

<table>
<thead>
<tr>
<th>PRES</th>
<th>data identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>WASHINGTON</td>
<td>President-identifier (usually an abbreviation of the last name)</td>
</tr>
<tr>
<td>WASHINGTON</td>
<td>last-name</td>
</tr>
<tr>
<td>GEORGE</td>
<td>first-name</td>
</tr>
<tr>
<td></td>
<td>middle-initial</td>
</tr>
<tr>
<td>FEBRUARY</td>
<td>month</td>
</tr>
<tr>
<td>22</td>
<td>day</td>
</tr>
<tr>
<td>1732</td>
<td>year</td>
</tr>
<tr>
<td>VIRGINIA</td>
<td>state-born-in</td>
</tr>
<tr>
<td>VIRGINIA</td>
<td>state-identifier (identifier name for the state born in)</td>
</tr>
<tr>
<td>6 FT. 2 IN.</td>
<td>height</td>
</tr>
<tr>
<td>FEDERALIST</td>
<td>party</td>
</tr>
<tr>
<td></td>
<td>college</td>
</tr>
<tr>
<td>ENGLISH</td>
<td>ancestry</td>
</tr>
<tr>
<td>EPISCOPAL</td>
<td>religion</td>
</tr>
<tr>
<td>3</td>
<td>#-of-occupations</td>
</tr>
<tr>
<td>SURVEYOR</td>
<td>occupations</td>
</tr>
<tr>
<td>FARMER</td>
<td></td>
</tr>
<tr>
<td>SOLDIER</td>
<td></td>
</tr>
<tr>
<td>DECEMBER</td>
<td>month</td>
</tr>
<tr>
<td>14</td>
<td>day</td>
</tr>
<tr>
<td>1799</td>
<td>year</td>
</tr>
<tr>
<td>PNEUMONIA</td>
<td>cause-of-death</td>
</tr>
<tr>
<td>AUGUSTINE</td>
<td>father-name</td>
</tr>
<tr>
<td>MARY</td>
<td>mother-name</td>
</tr>
<tr>
<td>1</td>
<td>#-of-marriages</td>
</tr>
</tbody>
</table>

* All the data in PRESFILE, including identifiers, were restricted to ten character representation thus resulting in abbreviations of some names.
MARTHA
wife

JANUARY
month

6
day

1759
year

0
number-of-children

2
#-of-elections-won

E1789
election-identifiers

E1792

2
#-of-administrations-headed

A1
administration-identifiers

A2

4
#-of-cabinets

C1
cabinet-identifiers

C2

C3

C4

(number of occurrences of marriage-data is determined by value of #-of-marriages)

(number of elections is determined by value of #-of-elections-won)

(number of administrations is determined by value of #-of-administrations-headed.)

(number of cabinets is determined by value of #-of-cabinets.)

Note that any fields which are not defined (such as middle initial and college in this example) are represented by ten blanks.

The format of the data following an ELECTION identifier is given below:

ELECTION
data identifier
E1789
election-identifier
1789
election-year
WASHINGTON
winner
FEDERALIST
winning-party
69
winning-votes
1
#-of-opponents
ADAMS
opponent

opponent-data

FEDERALIST
opponent-party

opponent-votes

(number of occurrences of opponent-data is determined by value of #-of-opponents.)
The format of the data following an ADMIN identifier is given below:

```
ADMIN    data-identifier
Al       administration-identifier
APRIL    month
30       day
1789     year
WASHINGTON  president-identifier
1         #-of-vice-presidents
JOHN     first-name
ADAMS    surname
          vice-president-name
          (number of occurrences of vice-president name is determined by value of #-of-vice-presidents.)
2         #-of-states admitted
VERMONT  state-name
VERMONT  state-identifier
          state-data
          (Number of occurrences of state-data is determined by value of #-of-states-admitted.)
KENTUCKY state-name
KENTUCKY state-identifier
          state-data
```

The format of the data following a CONGRESS identifier is given below:

```
CONGRESS  data-identifier
CI        Congress-identifier
2         #-of-Senate-parties
```
FEDERALIST  17 Senate-partySenate-data
           number-of-Senators (number of occurrences of Senate-data is determined by value of #-of-Senate-parties)

ANTI-FED  9 Senate-partySenate-data
           number-of-Senators
           #-of-House-parties

FEDERALIST  38 House-partyHouse-data
           number-of-members (Number of occurrences of House-data is determined by the value of #-of-House parties.)

ANTI-FED  26 House-partyHouse-data
           number-of-members

The format of the data following a STATES identifier is given below.

STATES data-identifier
ALABAMA state-identifier
ALABAMA state-name
   1814 year-admitted
MONTGOMERY capital
   51609 area
   29 area rank (as compared to all 49 other states)
   3556000 population
   21 population-rank (as compared to all 49 other states)
   10 number-of-House-votes
   4 #-of-major-cities
BIRMINGHAM city-name
   340887 city-population city-data
MOBILE city-name
   202779 city-population city-data
   (The number of occurrences of city-data is determined by the value of #-of-major-cities.)

MONTGOMERY city-name
   134393 city-population city-data
HUNTSVILLE city-name
   123519 city-population city-data
Notice that state-identifier state-name are usually the same. The major difference between the two is that blanks and special characters are allowed in a state-name but not for state-identifier. For example, "NEW YORK" is the name of the state, but "NEWYORK" is the identifier for the state. (This was a restriction imposed by the PRESFILE format rather than the DBMS.) The complete listing of PRESFILE can be found in Appendix A.

2.4 A Data Structure for the Presidential Data Base

Now that all the data available to the problem is known, an attempt can be made to structure it.

First off, a record type can be defined to correspond with each of the five possible data group types (PRES, ELECTION, ADMIN, CONGRESS and STATES). Some of the data in these groups are redundant (occur in more than one group) and so an attempt to eliminate redundancy will be made by specifying relationships between records. For example, within the group of data describing a President, the state he was born in and the identifier for that state are given. This data could be eliminated altogether if a relation was set up between the President record and State record specifying that the particular President was born in that State.

After eliminating redundant data the following relationships are given between record types.

![Data Structure Diagram]

**Diagram Description:**
- **PRESIDENT** node has relationships with:
  - **ELECTION** node (1:M) with identifier
  - **CONGRESS** node (1:M) with identifier
  - **STATE** node (1:M) with identifier and population
  - **ADMINISTRATION** node (1:M) with states admitted identifier
- **ELECTION** node has identifier
- **CONGRESS** node has identifier
- **STATE** node has identifier
- **ADMINISTRATION** node has states admitted identifier
Note that all record types are ordered by identifier for retrieval purposes. STATE is also ordered by population size.

The remaining task to be performed is to identify the data within each record type and identify each piece of data as an "item," "group," "repeating item" or "repeating group." Though the contents of the record types are very similar to the contents of the data groups specified in the PRESFILE it will be presented here for the sake of clarity.

President Record --

<table>
<thead>
<tr>
<th>Data represented</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>President-identifier</td>
<td>item</td>
</tr>
<tr>
<td>First-name (item)</td>
<td></td>
</tr>
<tr>
<td>Middle-initial (item)</td>
<td></td>
</tr>
<tr>
<td>Surname (item)</td>
<td></td>
</tr>
<tr>
<td>month (item)</td>
<td></td>
</tr>
<tr>
<td>day (item)</td>
<td></td>
</tr>
<tr>
<td>year (item)</td>
<td></td>
</tr>
<tr>
<td>Height</td>
<td>item</td>
</tr>
<tr>
<td>Party</td>
<td>item</td>
</tr>
<tr>
<td>College</td>
<td>item</td>
</tr>
<tr>
<td>Ancestry</td>
<td>item</td>
</tr>
<tr>
<td>Religion</td>
<td>item</td>
</tr>
<tr>
<td>month (item)</td>
<td></td>
</tr>
<tr>
<td>day (item)</td>
<td></td>
</tr>
<tr>
<td>year (item)</td>
<td></td>
</tr>
<tr>
<td>Father</td>
<td>item</td>
</tr>
<tr>
<td>Mother</td>
<td>item</td>
</tr>
<tr>
<td>#of-occupations</td>
<td>item</td>
</tr>
<tr>
<td>occupation</td>
<td></td>
</tr>
<tr>
<td>Wife (item)</td>
<td></td>
</tr>
<tr>
<td>month (item)</td>
<td></td>
</tr>
<tr>
<td>day (item)</td>
<td></td>
</tr>
<tr>
<td>year (item)</td>
<td></td>
</tr>
<tr>
<td>number-of-children (item)</td>
<td></td>
</tr>
<tr>
<td>Marriage-date*</td>
<td></td>
</tr>
<tr>
<td>Marriage-data</td>
<td></td>
</tr>
</tbody>
</table>

* Marriage-date is a group within a repeating group.
### Election Record --

<table>
<thead>
<tr>
<th>Data represented</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Election-identifier</td>
<td>item</td>
</tr>
<tr>
<td>Election-year</td>
<td>item</td>
</tr>
<tr>
<td>Winner</td>
<td>item</td>
</tr>
<tr>
<td>Winning-party</td>
<td>item</td>
</tr>
<tr>
<td>Winning-votes</td>
<td>item</td>
</tr>
<tr>
<td>Opponent (item)</td>
<td></td>
</tr>
<tr>
<td>Opponent-party (item)</td>
<td>OPponent-data</td>
</tr>
<tr>
<td>Opponent-votes (item)</td>
<td>repeating group</td>
</tr>
</tbody>
</table>

### Administration Record --

<table>
<thead>
<tr>
<th>Data represented</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration-identifier</td>
<td>item</td>
</tr>
<tr>
<td>month (item)</td>
<td></td>
</tr>
<tr>
<td>day (item)</td>
<td>inauguration-date</td>
</tr>
<tr>
<td>year (item)</td>
<td>group</td>
</tr>
<tr>
<td>first-name (item)</td>
<td></td>
</tr>
<tr>
<td>surname (item)</td>
<td>vice-president-name</td>
</tr>
<tr>
<td></td>
<td>repeating group</td>
</tr>
</tbody>
</table>

### Congress Record --

<table>
<thead>
<tr>
<th>Data represented</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Congress-identifier</td>
<td>item</td>
</tr>
<tr>
<td>party (item)</td>
<td></td>
</tr>
<tr>
<td>members (item)</td>
<td>Senate-data</td>
</tr>
<tr>
<td>party (item)</td>
<td>repeating group</td>
</tr>
<tr>
<td>members (item)</td>
<td>House-data</td>
</tr>
<tr>
<td></td>
<td>repeating group</td>
</tr>
</tbody>
</table>

### State Record --

<table>
<thead>
<tr>
<th>Data represented</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>State-identifier</td>
<td>item</td>
</tr>
<tr>
<td>State-name</td>
<td>item</td>
</tr>
<tr>
<td>Year-admitted</td>
<td>item</td>
</tr>
<tr>
<td>Capital</td>
<td>item</td>
</tr>
<tr>
<td>Area</td>
<td>item</td>
</tr>
</tbody>
</table>
State Record (Continued)

<table>
<thead>
<tr>
<th>Data represented</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area-rank</td>
<td>item</td>
</tr>
<tr>
<td>Population</td>
<td>item</td>
</tr>
<tr>
<td>Population-rank</td>
<td>item</td>
</tr>
<tr>
<td>Electoral-votes</td>
<td>item</td>
</tr>
<tr>
<td>City-name (item)</td>
<td>City-data</td>
</tr>
<tr>
<td>City-population (item)</td>
<td>repeating group</td>
</tr>
</tbody>
</table>
3. SPECIFYING THE DDL FOR A DATA STRUCTURE

Now that a data structure has been specified, this structure must be defined in DDL statements for processing by the DBMS. The conventions of translating the data structure into its equivalent DDL are given in the rest of this section. The only objects that can be defined by the DDL are RECORDS, ITEMS, REPEATING ITEMS, and SETS. All data described in the data structure must consequently be defined in these terms. Refer to Figure 1, "CARD FORMATS," and Figure 2, "CARD ORDERING RESTRICTIONS," for proper syntax of the DDL.

3.1 All data names defined by the user are restricted to a six character representation. For example, in the Presidential Data Base Example "President-identifier" and "college" were two names representing items types in the data structure. Because of the six character restriction these names must be truncated or abbreviated. "President-identifier" could be called IDENT and "college" COLEGE. Names of record types, however, must be unique for the entire system. This means that two item types could have the same name "IDENT" as long as they were contained in two different record types. Two record types of the name "PRES" would be illegal.

3.2 Records and Items

Since records and items are objects allowed in the DDL terminology, the records and items defined in the data structure may be translated directly into DDL statements. For example, the data for the Election record could be defined as such:

```
RECORD ELECT
ITEM IDENT CHAR 10
ITEM YEAR INTEG 31
ITEM WINNER CHAR 10
ITEM PARTY CHAR 10
ITEM VOTES INTEG 15
```

The repeating group in the Election record must be handled in a special way and will be described later. Notice that each item is defined to be either "CHAR" or "INTEG." This specifies what type of data is contained within the item; a character string or an integer number, respectively. There are six possible types of allowable data:

```
INTEG (integer number)
REAL (real number)
BINARY (binary number)
DBKEY (data base key value)
LOGIC (logical value)
CHAR (character string)
```
CARD FORMATS

RECORD card:

col 1 - 6  RECORD
col 8 - 13  record type name

ITEM card:

col 1 - 4  ITEM
col 8 - 13  item name
col 15 - 20  item type
legal types: INTEG, REAL, BINARY, DBKEY, LOGIC, CHAR

col 22 - 24  item size
legal size:  type     maxsize     units
            INTEG     31        bits
            REAL     63        bits
            BINARY   256       bytes
            DBKEY    -         always 4 bytes
            LOGIC    -         always 4 bytes
            CHAR     256       bytes

col 26 - 31  depended on item (optional)
col 33 - 35  max replication factor

SET card:

col 1 - 3  SET
col 8 - 13  set type name
col 15 - 20  order
legal orders: FIFO, LIFO, NEXT, PRIOR, IMMAT, SORTED

col 26 - 31  sort key (if order = SORTED)

OWNER card:

col 1 - 5  OWNER
col 8 - 13  owner record type name

MEMBER card:

col 1 - 6  MEMBER
col 8 - 13  member record type

NPAGES card:

col 1 - 6  NPAGES
col 22 - 24  data base size (in pages)

Figure 1
CARD ORDERING RESTRICTIONS

I. Record Descriptions

1. Items can only be defined in relation to the record they belong to (i.e. an item cannot be independent).

2. The items which make up a record type must be described by ITEM Cards and immediately follow the appropriate RECORD Card.

3. The order in which ITEM cards occur after the RECORD Card determines the order in which the data is physically stored in the record.

4. A repeating item is defined when the "max replication factor" field is assigned a value.

5. If a replication factor is given for an item, it is assumed to repeat.

6. A repeating item may only be the last item in a record.

7. The "depending on item" field for a repeating item designates an item (defined elsewhere in the record description) whose value specifies the number of occurrences of the repeating item (for a particular record occurrence).

8. A depending on item must be of type INTEG.

9. An item must be defined previously in the record before it can be used as a depending item in the record.

10. The depending on item is only checked for if a "max replication factor" is given.

II. Set Descriptions

1. There must be at least one owner card and one member card for each set.

2. If SYSTEM is given as an owner record type name, no other owner cards may be given for that set.

3. The owner card(s) for a set must come immediately after the SET card for the set.

4. The member card(s) for a set must come immediately after the OWNER card(s).

5. A record type must have been defined previously (by a RECORD card) before it can be used as an owner or member type.

Figure 2
II. Set Descriptions (Continued)

6. If a sort key is given for a set, it must be an item previously defined (in a record description) and be of the same type, size and displacement in all member record types.

7. A sort key item may only be of type CHAR or INTEG.
**IBM/360-370 STORAGE CHARACTERISTICS**

<table>
<thead>
<tr>
<th>Item type</th>
<th>Size defined in DDL</th>
<th>Allotted Storage Space</th>
<th>Range of values represented</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEG</td>
<td>1-31 bits</td>
<td>full word</td>
<td>-2147483648 thru +2147483647</td>
</tr>
<tr>
<td>REAL</td>
<td>1-31 bits</td>
<td>full word</td>
<td>± (16-65 thru 1663) **</td>
</tr>
<tr>
<td>REAL</td>
<td>32-63 bits</td>
<td>double word</td>
<td>± (16-65 thru 1663) **</td>
</tr>
<tr>
<td>BINARY</td>
<td>1 ≤ n ≤ 256 bytes</td>
<td>N words *</td>
<td>-2(32-N)-1 thru +2(32-N)-1</td>
</tr>
<tr>
<td>DBKEY</td>
<td>(always 4 bytes)</td>
<td>full word</td>
<td>(not relevant)</td>
</tr>
<tr>
<td>LOGIC</td>
<td>(always 4 bytes)</td>
<td>full word</td>
<td>True or False</td>
</tr>
<tr>
<td>CHAR</td>
<td>1 ≤ n ≤ 256 bytes</td>
<td>N words *</td>
<td>0 thru 4·N characters</td>
</tr>
</tbody>
</table>

* Where $N = f(\frac{n+3}{4})$ [f rounds the expression to the lowest integer]
  For example, if $n=10$: $N=f(\frac{10+3}{4})=3$ full words.

** This range is approximately $0.5397605 \times 10^{-78}$ through $0.7237005 \times 10^{76}$ for both single (full word) and double precision (double word). Double precision, however, allows for seven additional digits of accuracy over the single precision equivalent. Of course the representation of the zero value 0.0 can be represented as a real number in addition to these value ranges.

Table 1
The maximum size of the data value in each item is specified as either "10", ten bytes (characters) or "15" and "31" bits (integer). Table 1 specifies the ranges of allowable size for each of the six above data types.

3.3 Item Size

The DDL allows the user to specify a size for each item defined. Regardless of the size given, however, the data must be stored as a full word or some integer multiple of full words. Any unused space in the full word will be packed with an appropriate filler (e.g., blanks if the value is a character string and zeros if the value is an integer.) If a size of 6 bits were given for an item defined to be of type INTEG, each occurrence of this item would be reserved a full word of storage. The size here is only of relevance to the user as it designates some restriction to the range of values which should be accepted by the user's program. The DBMS makes no distinction if the size is 15 or 31 bits (in the case of integers) since an entire fullword is saved to store any integer value.

Treatment of item size for character strings (type CHAR) or bit strings (BINARY) is a bit different than that of type INTEG. Table 1 presents a method of calculating the number of full words of storage assigned to each occurrence of an item (as defined by its type and size). It also presents the range of values allowed for a particular item (as defined by its type and size). Note that the range of values is directly dependent on the allotted storage space.

3.4 Handling Relationships Among Records

It is a simple matter to define record relationships in the data structure as SETs in the DDL.

For relationships defined as 1:1 or 1:M in the data structure (take the relationship between President records and Administration records for example):

```
President
  \__________/  \______________/
     |             |               |
President \__________| ELECTION-WON \ 1:M
   |                     |               |
Election
```
The President record will be denoted as PRES in DDL terminology, Election record as ELECT and the Elections-won relationship as PRESEN. The following algorithm can be used:

i. Choose a name for the relationship (it must be unique). (In this example the name used is PRESEN.)

ii. The record type at the tail of the relationship is designated as the OWNER record.

iii. The record type (or types) at the head of the relationship is designated as a MEMBER record.

iv. All the occurrences of member record types to a particular occurrence of an owner record can be ordered in some way if desired (See below.)

The DDL needed to represent the Elections-won relationships is given as:

```
SET PRESEN SORTED IDENT
OWNER PRES
MEMBER ELECT
```

The "SORTED IDENT" phrase means that all member record occurrences for a particular owner record occurrence will be sorted on the item, IDENT, within the member record. This ordering is for search and retrieval purposes.

For a relationship defined as M:M the task of specifying this in the DDL is a little more complex. (The relationship between Congress and President records will be used as the example.)

The President record is denoted as PRES in the DDL, the Congress record as CONGRS and the two relationships that will be defined are PRESCS and CONGPS. Also, another record type must be defined which will be called NUB.

The data structure will be changed to the following by the use of the NUB record:

```
\begin{tikzpicture}
  \node (PRES) {PRES};
  \node (PRESCS) [below of=PRES] {PRESCS \nodepart{second} 1:M};
  \node (NUB) [below of=PRESCS] {NUB};
  \node (CONGPS) [below of=NUB] {CONGPS \nodepart{second} 1:M};
  \node (CONGRS) [below of=CONGPS] {CONGRS};
  \draw (PRES) -- (PRESCS);
  \draw (PRESCS) -- (NUB);
  \draw (NUB) -- (CONGPS);
  \draw (CONGPS) -- (CONGRS);
\end{tikzpicture}
```
The DDL for this is specified in the same way as for two 1:M relationships:

```
SET     PRESCS   FIFO
OWNER   PRES     NUB
MEMBER  NUB      IMMAT
SET     CONGRS   IMMATT
OWNER   CONGRS   NUB
MEMBER  NUB      NUB
```

To specify an ordering relationship in the DDL, one item in the record type to be ordered must be chosen as the sort key. Take the example:

```
identifier

President
```

from the data structure. This specifies that the President record (PRES) is to be ordered by its identifier item (call it IDENT). This can be specified by the DDL in the following manner:

```
SET     PORDER   SORTED   IDENT
OWNER   SYSTEM   NUB      NUB
MEMBER  PRES     NUB      NUB
```

The SYSTEM record is a record type defined by the data base system.

3.5 Handling Groups

Since groups are not allowed in the data structure, a group must be defined in terms of "items". For example, "birthdate" in the President record must be defined in terms of month, day, year. Since another date group occurs in the President record (death date) more descriptive names must be chosen such as MONTHB, DAYB, and YEARB.
3.6 Handling Repeating Items

Repeating items are allowed by the DDL with some special declaration. Take the example of a repeating item as defined in the Presidential Data Structure which is "occupation" in the President record type. A maximum limit of how many times the item repeats must be specified in the DDL (in this case, 3). An item must be defined to maintain a count of the number of times the item repeats for any particular occurrence of the President record. In the DDL format, this information is presented as:

    ITEM NOOCC INTEG  15
    ITEM OCCUP CHAR  10  NOOCC  3

NOOCC is the item which maintains a count of the number of times OCCUP repeats. The "3" on the right specifies that, at most, a President can have three occupations listed.

Only one repeating item may be defined for a particular record type. Should there be more than one defined by the data structure, it should be thought of as a repeating group (which will be discussed next).

Last thing to be considered is that a repeating item is the last item to be defined (in the DDL format) for a particular record type.

3.7 Handling Repeating Groups

Any data defined in the data structure as a repeating group must be defined as a new record type in the DDL. For example, the repeating group, "marriage-date," in the President record must be defined as a new record type (call it MARRGE) and a 1:M relationship is formed between it and the PRES record (with PRES as the owner). The information in marriage-data can be presented by the following DDL:

    RECORD MARRGE
    ITEM  WIFE  CHAR  10
    ITEM  MONTM  CHAR  10
    ITEM  DAYM  INTEG  15
    ITEM  YEARM  INTEG  31
    ITEM  CHILDN  INTEG  15
    SET  PRESME  FIFO
    OWNER  PRES
    MEMBER  MARRGE
3.8 **Specifying Data Base Size**

One DDL statement that does not directly describe the contents of the data base, but rather the size of it is NPAGES. The value associated with it (must be an integer) specifies the number of pages to be allocated for storing the contents of the data base. It's true value can only be estimated, but additions and deletions from the data base must be taken into account. Exceeding the NPAGES limit will result in a warning message.

3.9 **Developing the DDL for the Presidential Example**

From the original data structure for this example a new one can be constructed to show the added record types (due to repeating groups) and their relationships to the rest of the data structure.

Figure 3 presents the new data structure to be described by the DDL. Table 2 presents the names given to the records, sets and items defined in the DDL.

Finally, Figure 4 shows the complete DDL as derived from the conventions in sections 3.1 to 3.8.

Note that all items which contain character information (e.g., FSTNAM and IDENT) are declared to have a length, 10 characters, equal to that specified in PRESFILE. Though some occurrences of items carrying integer values (e.g., CAPTAL and AREA in the STATE record) have lengths of 15 bits and others of 31 bits, all have a fullword of storage assigned to them (31 bits of storage, 32 including the sign bit).
Names used for Presidential Data Base
Records, Items and Sets

<table>
<thead>
<tr>
<th>Record</th>
<th>Item</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRES (president)</td>
<td>IDENT (identifier)</td>
<td>PRESEN (elections won)</td>
</tr>
<tr>
<td>FSTNAM (first name)</td>
<td>PRESAN (administrations headed)</td>
<td></td>
</tr>
<tr>
<td>INITIAL (initial)</td>
<td>PRESCS (congresses headed)</td>
<td></td>
</tr>
<tr>
<td>SURNAM (surname)</td>
<td>PRESS (state born in)</td>
<td></td>
</tr>
<tr>
<td>MONTHB (month born)</td>
<td>PRESME (marriages)</td>
<td></td>
</tr>
<tr>
<td>DAYB (day born)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YEARB (year born)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEIGHT (height)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARTY (political party)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLEGE (college)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANSTRY (ancestry)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RELIGN (religion)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MONTHD (month died)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAYD (day died)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YEARD (year died)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAUSE (cause of death)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FATHER (father's name)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOTHER (mother's name)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOOC (number of occupations)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCCUP (name of occupation)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| MARRGE (marriage) | WIFE (wife's name) |
| MOUTHM (month married) |                |
| DAYM (day married) |                |
| YEARM (year married) |                |
| CHILDN (number of children) |                |

Table 2
<table>
<thead>
<tr>
<th>Record</th>
<th>Item</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMIN (administration)</td>
<td>IDENT (identifier)</td>
<td>NUSTAT (new states added)</td>
</tr>
<tr>
<td></td>
<td>MONTH (month started)</td>
<td>CABINT (vice presidents)</td>
</tr>
<tr>
<td></td>
<td>DAY (day started)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YEAR (year started)</td>
<td></td>
</tr>
<tr>
<td>VPRES (vice president)</td>
<td>FSTNAM (first name)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SURNAM (surname)</td>
<td></td>
</tr>
<tr>
<td>ELECT (election)</td>
<td>IDENT (identifier)</td>
<td>ELECTO (opponents)</td>
</tr>
<tr>
<td></td>
<td>YEAR (election year)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WINNER (winner)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARTY (winning party)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VOTES (number of winning votes)</td>
<td></td>
</tr>
<tr>
<td>OPPON (opponent)</td>
<td>NAME (opponent's name)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARTY (opponent's party)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VOTES (opponent's number of votes)</td>
<td></td>
</tr>
<tr>
<td>CONGRS (congress)</td>
<td>IDENT (identifier)</td>
<td>SMEMBS (senate members)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HMEMBS (house members)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CONGPS (headed by)</td>
</tr>
<tr>
<td>SENATE (senate)</td>
<td>PARTY (party name)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NUMBER (number of seats)</td>
<td></td>
</tr>
<tr>
<td>HSEREP (house)</td>
<td>PARTY (party name)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NUMBER (number of seats)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 (Continued)
Record | Item | Set
---|---|---
STATE (state) | IDENT ((identifier) | CITIES (cities)
| NAME (state name) | |
| YEARAD (year admitted) | |
| CAPITAL (capital) | |
| AREA (area) | |
| ARANK (rank in area size) | |
| POP (population) | |
| PRANK (population rank) | |
| VOTES (electoral votes) | |
CITY (city) | NAME (city name) | |
| POP (population) | |

Ordering Relations

<table>
<thead>
<tr>
<th>Record</th>
<th>Set</th>
<th>Ordered on</th>
</tr>
</thead>
</table>
PRES | porder | IDENT (based on president's name) |
 ADMIN | aorder | IDENT (based on chronological order) |
 ELECT | eorder | IDENT (based on chronological order) |
 CONGRS | corder | IDENT (based on chronological order) |
 STATE | sorder | IDENT (based on state name) |
 STATE | stasiz | PRANK (based on population size) |

Table 2 (Continued)

-33-
RECORD PRES
ITEM IDENT CHAR 10
ITEM FSINAM CHAR 10
ITEM INITIAL CHAR 10
ITEM SURNAM CHAR 10
ITEM MONTNB CHAR 10
ITEM DAYB INTEG 15
ITEM YEARB INTEG 31
ITEM HEIGHT CHAR 10
ITEM PARTY CHAR 10
ITEM COLGE CHAR 10
ITEM ANSTRY CHAR 10
ITEM RELIGN CHAR 10
ITEM MONTND CHAR 10
ITEM DAYD INTEG 15
ITEM YEARD INTEG 31
ITEM CAUSE CHAR 10
ITEM FATHER CHAR 10
ITEM MOTHER CHAR 10
ITEM NOOCC INTEG 15
ITEM OCCUP CHAR 10 NOOCC 3
RECORD MARRGE
ITEM WIFE CHAR 10
ITEM MONTWM CHAR 10
ITEM DAYM INTEG 15
ITEM YEARM INTEG 31
ITEM CHILDN INTEG 15
RECORD ADMIN
ITEM IDENT CHAR 10
ITEM MONT CHAR 10
ITEM DAY INTEG 15
ITEM YEAR INTEG 31
RECORD VPRES
ITEM FSINAM CHAR 10
ITEM SURNAM CHAR 10
RECORD STATE
ITEM IDENT CHAR 10
ITEM NAME CHAR 10
ITEM YEARAD INTEG 31
ITEM CAPITAL CHAR 10
ITEM AREA INTEG 31
ITEM ARANK INTEG 15
ITEM POP INTEG 31
ITEM PRANK INTEG 15
ITEM VOTES INTEG 15
RECORD CITY
ITEM NAME CHAR 10
ITEM POP INTEG 31

Figure 4
The Presidential Data Base DDL
49 RECORD ELECT
50 ITEM IDENT CHAR 10
51 ITEM YEAR INT 31
52 ITEM WINNER CHAR 10
53 ITEM PARTY CHAR 10
54 ITEM VOTES INT 15
55 RECORD OPPON
56 ITEM NAME CHAR 10
57 ITEM PARTY CHAR 10
58 ITEM VOTES INT 15
59 RECORD CONORS
60 ITEM IDENT CHAR 10
61 RECORD SENATE
62 ITEM PARTY CHAR 10
63 ITEM NUMBER INT 15
64 RECORD HSREP
65 ITEM PARTY CHAR 10
66 ITEM NUMBER INT 15
67 RECORD NUB
68 SET PRESME FIFO
69 OWNER PRES
70 MEMBER MARRGE
71 SET PRESCS FIFO
72 OWNER PRES
73 MEMBER NUB
74 SET CONGPS IMMAT
75 .OWNER CONGRS
76 MEMBER NUB
77 SET PRESEN SORTED IDENT
78 .OWNER PRES
79 MEMBER ELECT
80 SET PRESAN SORTED IDENT
81 OWNER PRES
82 MEMBER ADMIN
83 SET NUSTAT SORTED YEARAD
84 OWNER ADMIN
85 MEMBER STATE
86 SET CITIES SORTED POP
87 OWNER STATE
88 MEMBER CITY
89 SET CABINT FIFO
90 .OWNER ADMIN
91 MEMBER VPRES
92 SET SMEMBS SORTED NUMBER
93 .OWNER CONGRS
94 MEMBER SENATE
95 SET HMEMBS SORTED NUMBER
96 OWNER CONGRS
97 MEMBER HSREP

Figure 4 (Continued)
98  SET SIASIZ SORTED PRANK
99  OWNER SYSTEM
100 MEMBER STATE
101 SET PORDER SORTED IDENT
102 OWNER SYSTEM
103 MEMBER PRES
104 SET SORDER SORTED IDENT
105 OWNER SYSTEM
106 MEMBER STATE
107 SET CORDER SORTED IDENT
108 OWNER SYSTEM
109 MEMBER CONGRS
110 SET EORDER SORTED IDENT
111 OWNER SYSTEM
112 MEMBER ELECT
113 SET AORDER SORTED IDENT
114 OWNER SYSTEM
115 MEMBER ADMIN
116 SET ELECTO SORTED VOTES
117 OWNER ELECT
118 MEMBER OPPON
119 SET PRESS IMMAT
120 OWNER STATE
121 MEMBER PRES
122 NPAGES 20

Figure 4 (Continued)
4. CREATING AND INITIALIZING THE DATA BASE

After the DDL has been specified and stored in some way (either in a line file in the computer or on cards) it can be used to generate the Data Base Table File (DBTF). Once this is done the Data Base File (DBF) can be initialized (a preliminary action needed before any data can be stored in it).

4.1 Creation of DDL, DBTF and DBF files

As stated above, if the DDL for the data base is not punched on cards, it can be kept in a line file* in the computer. The DBTF and DBF files, however, must be sequential files*. If one DBMS is being developed it is a common convention to call the DDL file "DDL," the DBTF file, "DBTF" and DBF, "DBF." These names will be used in the following examples.

4.2 Generation of the DBTF*

First the DBTF file must be created:

$CREATE DBTF TYPE=SEQ

and then the following program should be run:

$RUN SELW:DLA 5=DDL 8=DBTF

assuming that "DDL" is the name of the file containing the DDL for the DBMS. Two additional parameters may be used in conjunction with this program: 6=fdbname and 7=fdbname. 6 designates where the printed output from the program should be printed (it defaults to *SINK*). 7 designates file or device where block data generated from the program should be generated (it defaults to *SINK*).

4.3 Initialization of the DBF*

After the DBTF has successfully been generated and the DBF has been created by:

$CREATE DBF TYPE=SEQ

the data base file can be initialized by the following program:

$RUN SELW:DBIN \underline{\text{DBTF}} \underline{\text{DBF}}

Once this has been done the data base is ready to be accessed for storage and retrieval of data.

* The file types, commands, and programs used are particular to the Michigan Terminal System (MTS).
5. CODING CONVENTIONS FOR DATA BASE ACCESS PROGRAMS

Some conventions are common to any programs used to access the data base whether it is to store data, modify data stored, or retrieve data. (All examples of programs in this paper will be written in FORTRAN).

5.1 Common Areas and Block Data

All the RECORD, ITEM and SET names defined in the DDL (e.g., PRES, IDENT and PORDER in the Presidential Example) have to be represented as character strings (i.e., 'PRES ', 'IDENT ', and 'PORDER ', respectively) in order to be used by the Data Base Control System (DBCS) subroutines* for access programs. For example, a call to a DBCS routine could be stated as:

```
CALL FMSK('PORDER ',BUF,IERR)
```

An alternative to enclosing all DDL names in quotes is to define a constant to hold each character string:

```
INTEGER PORDER(2)
DATA PORDER/'PORD'/'ER '/
```

(Note that the constant, PORDER must be of dimension '2' to hold six characters.** Now the same CALL statement can be written as

```
CALL FMSK(PORDER,BUF,IERR)
```

which saves a considerable bit of coding since the constant will probably be used several times in a program.

A lot of coding effort can be saved by usage of the FORTRAN COMMON statements and BLOCK DATA facility. All of the constants' value assignments and type assignments (e.g., INTEGER) can be given by the BLOCK DATA subprogram and COMMON statements which can be used to relate these constants in BLOCK DATA to the main program and any subroutines.

The information in the BLOCK DATA subprogram and COMMON areas are usually required for any subsequent programs and can be saved for this purpose also.

An example format of a BLOCK DATA subprogram is given in Figure 5. The COMMON area, NAMS, contains all the constant names. The second COMMON area, PARS, contains the buffer variables used in common with several programs and subroutines.

* Those subroutines described in ISDOS Working Paper No. 88
** If the DBMS is to be written to conform to ANSI standards, the DATA statement should specify Hollerith formats:

```
DATA PORDER/4HPORD,4HER /
```
Figure 5 Block data example
Figure 5 Block data example
5.2 Buffer Variables

Buffer variables are variables used to temporarily store data values as they pass to, or from, the data base. As specified in the COMMON area, PARS, NUM is a buffer variable used to hold all integer numerical data. BUF is a dimension 3 buffer variable which is used to hold all the ten character strings as read from PRESFILE. Note that in the Presidential Example all data is read in as ten characters and stored in BUF. Then the data is stored in the data base as a character string (ten characters) or converted to an integer and stored as a number. In DBMS, where many different sizes of data (measured in terms of fullwords), or types of data (logical, real, etc.) are to be read, a buffer variable must be defined for each different size and type. For example, if strings of 10 characters and 30 characters were to be manipulated the two buffers to handle these strings could be defined as:

```
INTEGER    BUF1(3),BUF2(8)
```

In any case, Table 1 given in Section 3, can be used as an aid in determining the dimension of the buffer variables to handle large character and bit strings. To define a buffer variable to hold double word size:

```
REAL* 8    RBUF
```

5.3 Access of the DBF and DBTF

Reference can be made to the Data Base File and Data Base Table File in a DBCS subroutine through assignment to logical IO units. For example, if the assignment of 2=DBF and 3=DBTF was made on the MTS $RUN command, the files could be referenced in the following manner:

```
CALL OPEN(2,3,100,IERR)
```
5.4 General Program Formats

Regardless of the purpose of the program accessing the database a general format is inherent. The figure below presents the general format for main programs.

```
COMMON/NAMS/
  ...
  ...

INTEGER ARRAY
  ...  \{ any data declarations particular to the main program \}
  ...

CALL REPI
CALL NEWREP(.FALSE.)
CALL OPEN(2,3,100,IERR)
  \{ program code \}

CALL CLOS(0,ARRAY,IERR)
END
```

REPI and NEWREP perform initialization required before using the report subsystem subroutines (to be described later). OPEN specifies the DBF and DBTF files to be accessed by the program and CLOS closes these files properly. The figure below presents the general format for subroutines used in accessing the data base.

```
SUBROUTINE routine-name
  COMMON /NAMS/
    ...
    ...

  \{ any data declarations particular to the subroutine \}
  \{ code \}

RETURN
END
```
6. POPULATING THE DATA BASE

One major function of the DBMS is to populate the data base (i.e., store occurrences of each type of SET, RECORD and ITEM defined by DDL). There are several subroutines described in ISDOS Working Paper No. 88 and in the PSA software documentation which are helpful (if not necessary) in accomplishing this.

6.1 Reading Data from the Input File

Before any data can be stored in the data base it must first be read from some input file or device. The QOPEN, QREAD and QRCLOS routines can aid in reading the data. QOPEN specifies which file the data is to be read from, and the size of the logical record to be read by QREAD. For example,

    CALL QOPEN(1,10,100)

specifies that the file used is assigned to Logical I/O unit 1 and the logical record size is ten characters within a block of size 100 (this obviously means that 10 records per block will be read as is the case of PRESFILE shown in Appendix A.)

QREAD reads in a logical record, stores the data and sets a switch should an end-of-file be encountered. For example,

    CALL QREAD(10,BUF,EOF)

specifies that the first 10 characters (which in this case is all the characters) of the logical record be stored in the variable BUF. EOF is a logical variable which will be set to .TRUE. if an end-of-file is encountered. Since all data is read in a character format the subroutine CTOI can be used to convert character to integer. For example,

    CALL CTOI(BUF,1,10,NUM)

will convert the character form of the number stored in BUF (starting in column 1 and which is ten characters long) to integer form and place the result in NUM. This is necessary when integer data is actually to be stored in the data base. If a data base "item" was defined to contain character information (in the DDL) then only character data can be stored in it. Likewise with integer or real numbers as data.

Finally QRCLOS closes the file designated in QOPEN. For example,

    CALL QRCLOS

accomplishes this.
6.2 Storing Data

Now that data can be read by the access program in some form (character and integer are the forms concerning this application) it must be stored in the data base as an "item" for later retrieval. The following routines described in this section are considered the basic subset of subroutines (from ISDOS Working Paper No. 88) needed to populate a data base.*

- FMSK Used to find a particular member record occurrence in a particular set.
- CR Used to create record occurrences.
- SFR Used to store data in a particular "item" of a particular record.
- AMS
- SRM
- SOM Used to store data in a particular set.
- SMOVE Used to define relationships between a record occurrence and a particular set.

6.3 FMSK Find Member of a set based on Sort Key

This routine was used (in the Presidential example) to check that a record occurrence based on the record identifier (sort key) for a particular record type had not been previously stored in the data base. For example, if BUF contained the identifier "WASHINGTON" the call

CALL FMSK(PORDER,BUF,IERR)

would check member record occurrences of the set PORDER to see that an occurrence has the sort key value contained in BUF. (The reader may note that this operation is consistent with the DDL specified for the Presidential example, i.e., PORDER is a defined set sorted on the item IDENT in the PRES record type and "WASHINGTON" is the value IDENT will take on for the PRES record occurrence for George Washington.) If a record occurrence can not be found that has "WASHINGTON" as an identifier, IERR will take on the value -1. Otherwise, the record occurrence found will be made the current member of the set PORDER. This makes it possible to store or retrieve data from the record occurrence. If IERR has the value -1, the record occurrence does not exist and a record occurrence must be created before any data can be stored in it.

6.4 CR Create a Record occurrence

This routine is used to create an occurrence of a particular record type (e.g., PRES) so that data may be stored in it and relationships to other record occurrences may be defined.

* The subroutine SMOVE description can be found in the PSA software documentation package.
For example, an occurrence of a PRES record type can be created by the call:

    CALL CR(PRES,KEY,IERR)

The data base key value for this occurrence is returned in KEY and any error codes are returned in IERR.

6.5 SFR    Set Field in the current Record

This routine is used to store data in a particular item (field) in a record occurrence. This would be used to store the character string "WASHINGTON" contained in BUF in the item called IDENT in a PRES record occurrence. For example,

    CALL SFR(IDENT,PRES,BUF,IERR)

would store the value of BUF in the item IDENT of the current PRES record occurrence. Any error code would be returned in IERR. It is important that BUF and IDENT are defined to hold the same type of data (i.e., INTEGER, CHARACTER, REAL, LOGICAL, etc.)

6.6 AMS    Add a particular record occurrence as a Member of a Set

Now that IDENT has a value the record occurrence may be added to the set PORDER as a member. (This could not have been done previously because PORDER is sorted on IDENT values.) For example,

    CALL AMS(PORDER,PRES,IERR)

adds the current PRES record occurrence to the set PORDER and makes this record occurrence the current member of the set. The record occurrence is added to the set according to any ordering criteria defined in the DDL (i.e., FIFO, SORTED, etc.).

At this point the particular record occurrence is the current member of the record type PRES and the current member record occurrence of the set PORDER.

6.7 SRM    Set Record occurrence based on current Member of set

In the case where the member record occurrence was found, this routine can set the current member of a set to be the current record occurrence of a particular record type. For example,

    CALL SRM(PORDER,IERR)

would set the current member record occurrence of the set PORDER to also be the current record occurrence of its particular record type (which in this case is the PRES record type).

At this point the particular record occurrence is the current record occurrence of the record type PRES and the current member record occurrence of the set PORDER.
6.8 SOM  Set the current Owner of a set, based on the current Member of a set.

This routine is used to set up relationships between two particular record occurrences. Before a record occurrence is designated as a member of a set its owner must be defined. (In ordering relationships this is not necessary since there is only one owner record occurrences; SYSTEM). For example,

```
CALL SOM(PRESME,PORDER,IERR)
```

makes the current member record of the set PORDER to be the current owner record of the set PRESME (marriage relationships). Once a marriage record (MARRGE) is created it can be related to a particular owner record by simply adding it to the PRESME set. For example,

```
CALL AMS(PRESME,MARRGE,IERR)
```

designates that the current MARRGE record is a member of the set PRESME. Its related owner record is defined by the previously used SOM routine.

6.9 SMOVE  String MOVE

This routine is used to store data in a repeating item. Using the example of the repeating item defined in the Presidential DDL:

```
ITEM NOOCC INTG 15
ITEM OCCUP CHAR 10 NOOCC 3
```

Remember that the value of NOOCC is the number of occurrences of the repeating item OCCUP for any given occurrence of the PRES record. When NOOCC=3 there must be three occupations stored (each as a 10 character name). Also note that 3 is the maximum number of occurrences of the repeating item. Assuming that the number of occurrences of OCCUP (value of NOOCC) is supplied by PRESFILE before the values for OCCUP, (the format of PRESFILE shows this to be true) this data can be read and stored in the following manner.

```
CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)
CALL SFR(NOOCCE,PRES,NUM,IERR)
IF (NUM .EQ. 0) GOTO 111
PTR=1
INC=NUM
DO 11 I=1,INC
CALL QREAD(10,BUF,EOF)
CALL SMOVE(B,PTR,BUF,1,10)
11 PTR=PTR+10
CALL SFR(OCCUP,PRES,B,IERR)
111 :
```
First the value to be given NOOCC is read, converted to integer representation and stored. A test is made to see that if any occurrences of OCCUP exist and branching is done accordingly. The repeating item OCCUP can be considered to be one large storage location (big enough for 3 values) and so a pointer is used (PTR) to specify where in the storage location the data will be stored. Figure 6 shows the format of this location.

```
  1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
    \   \   \   \   \   \   \   \   \   \   \   \   \   \   \   \  
   OCCUP #1  OCCUP #2  OCCUP #3
```

Figure 6

INC is used to contain the NOOCC value in the DO statement. (This is a convention used throughout the coding process, rather than using NUM. Since NUM might be used within a DO loop, its value can be modified accidentally.) Next, each occurrence of OCCUP is read and stored in the variable B (dimension 8, to hold thirty characters). Note that PTR, INC. and B are all declared as integer variables. SMOVE is then used to store each occurrence of OCCUP in B at its proper position (as designated by PTR). It is also specified that all ten (10) characters are to be stored starting at the first (1). Note that PTR will take on values of 1, 11 and 21. Finally, after all available occurrences of OCCUP have been stored in B the contents may be transferred to OCCUP in the data base via the SFR subroutine.

6.10 Code to Populate the Presidential Data Base

Now that some of the basic operations have been described they can all be related together to show how the data in PRESFILE can be stored in the Presidential data base.

A main program is used to identify the type of data that will be read from PRESFILE (i.e., one of the five types: PRES, ELECTION, ADMIN, CONGRESS or STATES data). After this is determined the appropriate subroutine is called that will handle storage of this data (i.e., CRPS, CREN, CRAN, CRCS or CRSS, respectively).

To generate the object code for this code:

```bash
$RUN *FTN PAR=SOURCE=BLOCFILE+ADMIN+CONGRS+ELECT+MAIN+PRES+STATE LOAD=CODE
```
BLOCFILE, ADMIN, etc. are the names of the files containing the subroutines, block data and main program code for the population operation. Table 3 presents relationships between the data in PRESFILE, the subprogram handling it and the file holding the source code for the subprogram.

Table 3

<table>
<thead>
<tr>
<th>Date type in PRESFILE</th>
<th>Subprogram handling this data type</th>
<th>Source code file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type identifiers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRES data</td>
<td>main program</td>
<td>MAIN</td>
</tr>
<tr>
<td>ELECTION data</td>
<td>CRPS subroutine</td>
<td>PRES</td>
</tr>
<tr>
<td>ADMIN data</td>
<td>CREN subroutine</td>
<td>ELECT</td>
</tr>
<tr>
<td>CONGRESS data</td>
<td>CRAN subroutine</td>
<td>ADMIN</td>
</tr>
<tr>
<td>STATES data</td>
<td>CRCS subroutine</td>
<td>CONGRS</td>
</tr>
<tr>
<td>---</td>
<td>BLOCK DATA</td>
<td>STATE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLOCDATA</td>
</tr>
</tbody>
</table>

After the object code has been generated, the code can be linked up to the appropriate DBMS routines (e.g., QREAD, SFR, SMOVE, etc.) and executed by issuing:

$RUN CODE+SELW:DBLIB 1=PRESFILE 2=DBF 3=DBTF

The rest of this section presents listings of the source code for the above subprograms.
***** MAIN PROGRAM *****
* COMMON AREAS FOR MAIN *

COMMON/PARS/NUM,BUF,EOF
INTEGER NUM,BUF(3)
LOGICAL EOF

* DEFINE VARIABLES TO BE USED IN THIS SUBPROGRAM *

INTEGER ARRAY

* REPORT INITIALIZATION PROCEDURES *

CALL REPINT
CALL NEWREP(.FALSE.)

* OPEN THE DATA BASE FILE AND DATA BASE TABLE FILE
(LOGICAL I/O UNITS 2 AND 3 RESPECTIVELY) *

CALL OPEN(2,3,100,IERR)

* OPEN PRESFILE (LOGICAL I/O UNIT 1) TO BE READ BY BLOCKED
DATA I/O ROUTINES *

CALL QOPEN(1,10,100)

* READ FIELD IN PRESFILE (SHOULD CONTAIN A DATA TYPE
IDENTIFIER) *

CALL QREAD(10,BUF,EOF)

* END PROCESSING SHOULD AN END-OF-FILE BE ENCOUNTERED *

IF (EOF) GOTO 999

* TEST TO SEE IF THE IDENTIFIER SPECIFIES PRESIDENTIAL DATA *

J=ISCOMP(BUF,1,'PRES',1,4)
IF (J .NE. 0) GOTO 98

* IF SJ, CALL ROUTINE CRPS TO HANDLE THIS TYPE OF DATA *

CALL CRPS
GOTO 99

* TEST TO SEE IF THE IDENTIFIER SPECIFIES ELECTION DATA *

J=ISCOMP(BUF,1,'ELECTION',1,8)
IF (J .NE. 0) GOTO 97

* IF SO, CALL ROUTINE CREN TO HANDLE THIS TYPE OF DATA *

CALL CREN
GOTO 99

* TEST TO SEE IF THE IDENTIFIER SPECIFIES ADMINISTRATION DATA *

J=ISCOMP(BUF,1,'ADMIN',1,5)
IF (J .NE. 0) GOTO 96
* IF SO, CALL ROUTINE CRAN TO HANDLE THIS TYPE OF DATA *
CALL CRAN
GOTO 99

* TEST TO SEE IF THE IDENTIFIER SPECIFIES CONGRESS DATA *
J=ISCOMPL(BUF,1,'CONGRESS',1,8)
IF (J .NE. 0) GOTO 95

* IF SO, CALL ROUTINE CRCS TO HANDLE THIS TYPE OF DATA *
CALL CRCS
GOTO 99

* TEST TO SEE IF THE IDENTIFIER SPECIFIES STATE DATA *
J=ISCOMPL(BUF,1,'STATES',1,6)
IF (J .NE. 0) GOTO 94

* IF SO, CALL ROUTINE CRSS TO HANDLE THIS TYPE OF DATA *
CALL CRSS
GOTO 99

* ILLEGAL IDENTIFIER ENCOUNTERED, PRINT WARNING MESSAGE AND STOP *

CALL BUFBLD(1,1,31,31,HWARNING-RECORD-TYPE)
CALL BUFBLD(10,1,10,BUF)
CALL PBUF(2,0,TRUE.)

* CLOSE PRESFILE *

CALL WRCLOS

* CLOSE THE DATA BASE *

CALL CLOSO(0,ARRAY,IERR)
END
SUBROUTINE CRPS

***** THIS ROUTINE HANDLES ALL DATA FOLLOWING A
PRESIDENT DATA TYPE IDENTIFIER *****

* COMMON AREAS FOR CRPS *

COMMON/NAMS/PRES,IDENT,FSTNAM,INITAL,SURNAM,
& MONTHS,DAYY, YEARB,HEIGHT, PARTY, COLEGE,
& ANSTRY, RELIGN, MOUTH, DAYY, YEARD, CAUSE,
& FATHER, MOTHER, NOOCC, OCCUP, MARRGE, WIFE,
& MONTHM, DAYM, YEARM, CHILDN, ADMIN, MOUTH,
& DAY, YEARM, VRES, STATE, NAME, YEARM,
& CAPITAL, AREA, ARANK, POP, PRANK, VOTES,
& CITY, ELECT, WINNER, UPON, CONGRS, SENATE,
& NUMBER, HSEREP, PRESME, PRESSCS, PRESEN,
& PRESAN, NUSTAT, CITIES, CABINT, SMEMBS, HMEMBS,
& STASIZ, PORDER, SORDER, CORDER, EORDER, AORDER,
& ELECTO, PRESS, NUB, CONGPS

INTEGER PRES(2), IDENT(2), FSTNAM(2), INITAL(2), SURNAM(2),
& MONTHH(2), DAYB(2), YEARB(2), HEIGHT(2), PARTY(2), COLEGE(2),
& ANSTRY(2), RELIGN(2), MOUTH(2), DAY(2), YEARD(2), CAUSE(2),
& FATHER(2), MOTHER(2), NOOCC(2), OCCUP(2), MARRGE(2), WIFE(2),
& MONTHM(2), DAYM(2), YEARM(2), CHILDN(2), ADMIN(2), MOUTH(2),
& DAY(2), YEARM(2), VRES(2), STATE(2), NAME(2), YEARM(2),
& CAPITAL(2), AREA(2), ARANK(2), POP(2), PRANK(2), VOTES(2),
& CITY(2), ELECT(2), WINNER(2), UPON(2), CONGRS(2), SENATE(2),
& NUMBER(2), HSEREP(2), PRESME(2), PRESSCS(2), PRESEN(2),
& PRESAN(2), NUSTAT(2), CITIES(2), CABINT(2), SMEMBS(2), HMEMBS(2),
& STASIZ(2), PORDER(2), SORDER(2), CORDER(2), EORDER(2), AORDER(2),
& ELECTO(2), PRESS(2), NUB(2), CONGPS(2)

COMMON/PARS/NUM, BUF, EOF

INTEGER NUM, BUF(3)

LOGICAL EOF

* DEFINE VARIABLES TO BE USED IN THIS SUBPROGRAM *

INTEGER B(8), PTR

* READ THE PRESIDENT IDENTIFIER FROM PRESFILE *

CALL QREAD(10, BUF, EOF)

* TEST TO SEE IF THE PRESIDENT IS ALREADY IN THE
DATA BASE *

CALL FMSK(PORDER, BUF, IERR)

IF (IERR .GT. -1) GO TO 110

* IF SO, GO TO 110; IF NOT, CREATE A RECORD OCCURRENCE
FOR HIM AND STORE THE IDENTIFIER *

CALL CRIPRES, KEY, IERR)

CALL SFPATIDENT, PRES, BUF, IERR)

* ADD THIS RECORD OCCURRENCE TO THE SET 'PORDER' *

CALL AMS(PORDER, PRS, IERR)
* MAKE THE RECORD THE CURRENT RECORD OCCURRENCE
  OF THE RECORD TYPE 'PRES' *

110 CALL SRF(PORDER, IERR)

* READ THE PRESIDENT'S LAST NAME AND STORE IT IN
  'SURNAM' *

112 CALL QREAD(10, BUF, EOF)
CALL SFR(SURNAM, PRES, BUF, IERR)

* READ HIS FIRST NAME AND STORE IT IN 'FRSTNAM' *

CALL QREAD(10, BUF, EOF)
CALL SFR(FRSTNAM, PRES, BUF, IERR)

* READ HIS MIDDLE INITIAL AND STORE IT IN 'INITIAL' *

CALL QREAD(10, BUF, EOF)
CALL SFR(INITIAL, PRES, BUF, IERR)

* READ HIS MONTH OF BIRTH AND STORE IT IN 'MONTHB' *

CALL QREAD(10, BUF, EOF)
CALL SFR(MONTHB, PRES, BUF, IERR)

* READ HIS DAY OF BIRTH, CONVERT TO NUMBER, AND
  STORE IN 'DAYB' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
CALL SFR(DAYB, PRES, NUM, IERR)

* READ HIS YEAR OF BIRTH, CONVERT TO NUMBER, AND
  STORE IN 'YEARB' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
CALL SFR(YEARB, PRES, NUM, IERR)

* READ STATE NAME(AND STATE IDENTIFIER) BORN IN.
  ONLY STATE IDENTIFIER NEED BE USED TO FORM THE
  RELATIONSHIP BETWEEN 'PRES' AND 'STATE' RECORDS *

CALL QREAD(10, BUF, EOF)
CALL QRFAD(10, BUF, EOF)

* TEST TO SEE IF THE STATE IS ALREADY IN THE DATA BASE *

CALL FMSK(SORDER, BUF, IERR)
IF (IERR .GT. -1) GOTO 101

* IF SO, GO TO 101; IF NOT, CREATE A RECORD OCCURRENCE
  FOR IT AND STORE STATE IDENTIFIER IN IDENT *

CALL CR(STATE, KEY, IERR)
CALL SFR(IDENT, STATE, BUF, IERR)
* ADD STATE RECORD OCCURRENCE TO SET 'SORDER' *

CALL AMS(SORDER, STATE, IERR)

* FORM THE RELATIONSHIP (STATE OF BIRTH) BETWEEN PRESIDENT AND STATE *

CALL SUM(PRESS, SORDER, IERR)
CALL AMS(PRESS, PRES, IERR)

* READ HIS HEIGHT AND STORE IN 'HEIGHT' *

CALL QREAD(10, BUF, EOF)
CALL SFR(HEIGHT, PRES, BUF, IERR)

* READ HIS PARTY AND STORE IN 'PARTY' *

CALL QREAD(10, BUF, EOF)
CALL SFR(PARTY, PRES, BUF, IERR)

* READ HIS COLLEGE ATTENDED AND STORE IN 'COLEGE' *

CALL QREAD(10, BUF, EOF)
CALL SFR(COLEGE, PRES, BUF, IERR)

* READ HIS ANCESTRY AND STORE IN 'ANSTRY' *

CALL QREAD(10, BUF, EOF)
CALL SFR(ANSTRY, PRES, BUF, IERR)

* READ HIS RELIGION AND STORE IN 'RELIGN' *

CALL QREAD(10, BUF, EOF)
CALL SFR(RELIGN, PRES, BUF, IERR)

* READ NUMBER OF OCCUPATIONS, CONVERT TO NUMBER, AND STORE IN 'NOOCC' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
CALL SFR(NOOCC, PRES, NUM, IERR)

* TEST TO SEE IF ANY OCCUPATION DATA SHOULD BE STORED AS A REPEATING ITEM *

IF (NUM .EQ. 0) GOTO 111
PTR = 1
INC = NUM

* PACK THE OCCUPATIONS INTO A TEMPORARY BUFFER (BUF) BEFORE STORING IN DATA BASE *

DO 11 I = 1, INC

* READ AN OCCUPATION AND STORE IN 'BUF' *

CALL QREAD(10, BUF, EOF)
CALL SMOVE(8, PTR, BUF, 1, 10)
PTR = PTR + 10
* STORE THE CONTENTS OF 'BUF' INTO 'OCCUP' *

CALL SFR(OCCUP,PRES,B,IERR)

* READ MONTH DIED AND STORE IN 'MONTHD' *

CALL QREAD(10,BUF,EOF)
CALL SFR(MONTHD,PRES,BUF,IERR)

* READ DAY DIED, CONVERT TO NUMBER, AND STORE IN 'DAYD' *

CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)
CALL SFR(DAYD,PRES,NUM,IERR)

* READ YEAR DIED, CONVERT TO NUMBER, AND STORE IN 'YEARD' *

CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)
CALL SFR(YEARD,PRES,NUM,IERR)

* READ CAUSE OF DEATH AND STORE IN 'CAUSE' *

CALL QREAD(10,BUF,EOF)
CALL SFR(CAUSE,PRES,BUF,IERR)

* READ FATHER'S NAME AND STORE IN 'FATHER' *

CALL QREAD(10,BUF,EOF)
CALL SFR(FATHER,PRES,BUF,IERR)

* READ MOTHER'S NAME AND STORE IN 'MOTHER' *

CALL QREAD(10,BUF,EOF)
CALL SFR(MOTHER,PRES,BUF,IERR)

* READ NUMBER OF MARRIAGE DATA OCCURRENCES AND CONVERT TO NUMBER *

CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)

* SPECIFY RELATIONSHIP BETWEEN PRESIDENT AND MARRIAGE DATA *

CALL SOMPRESME,PORDER,IERR)
INC=NUM

* CREATE MARRIAGE RECORD AND STORE MARRIAGE DATA FOR EACH MARRIAGE DATA OCCURRENCE *

DO 12 I=1,INC
CALL CK(MARRGE,KEY,IERR)
CALL AMS(PRESME,MARRGE,IERR)

* READ NAME OF WIFE AND STORE IN 'WIFE' *

CALL QREAD(10,BUF,EOF)
CALL SFR(WIFE,MARRGE,BUF,IERR)
* READ MONTH OF MARRIAGE AND STORE IN 'MONTHM' *
CALL QREAD(10,BUF,EOF)
CALL SFRI(MONTHM,MARRGE,BUF,IERR)

* READ DAY OF MARRIAGE, CONVERT TO NUMBER, AND
STORE IN 'DAYM' *
CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)
CALL SFRI(DAYM,MARRGE,NUM,IERR)

* READ YEAR OF MARRIAGE, CONVERT TO NUMBER, AND
STORE IN 'YEARM' *
CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)
CALL SFRI(YEARM,MARRGE,NUM,IERR)

* READ NUMBER OF CHILDREN FROM MARRIAGE, CONVERT TO NUMBER,
AND STORE IN 'CHILDN' *
CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)
CALL SFRI(CHILDN,MARRGE,NUM,IERR)

* READ NUMBER OF ELECTIONS WON AND CONVERT TO NUMBER *
CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)

* IF PRESIDENT WON NO PRESIDENTIAL ELECTIONS, GO TO
131; OTHERWISE, SPECIFY RELATIONSHIP BETWEEN
PRESIDENT AND ELECTION DATA *

IF (NUM .EQ. 0) GOTO 131
CALL SOM(PRESEN,PORDER,IERR)
INC[NUM

* PROCESS EACH OCCURRENCE OF ELECTION DATA *
DO 13 I=1,INC

* RFAD ELECTION IDENTIFIER *
CALL QREAD(10,BUF,EOF)

* TEST TO SEE IF ELECTION IS ALREADY STORED IN DATA BASE *
CALL FMSK(EORDER,BUF,IERR)
IF (IERR .GT. -1) GOTO 14

* IF S0, GO TO 14; IF NOT, CREATE ELECTION RECORD
AND STORE IDENTIFIER IN 'IDENT' *
CALL CRI(ELECT,KEY,IERR)
CALL SFRI(IDENT,ELECT,BUF,IERR)

* ADD ELECTION RECORD TO SET 'EORDER' *
CALL A8S(EORDER,ELECT, IERR)
GOTO 13

* MAKE CURRENT MEMBER OF SET 'EORDER', CURRENT
RECORD OF TYPE 'ELECT' *

14 CALL SSM(EORDER, IERR)

* FORM RELATIONSHIP BETWEEN PRESIDENT AND ELECTION
RECORD *

13 CALL A8S(PRESEN,ELECT, IERR)

* READ NUMBER OF ADMINISTRATIONS AND CONVERT TO NUMBER *

131 CALL ORSAD(10,BUF, EOF)
CALL CT0I(BUF, I, 10, NUM)

* SPECIFY RELATIONSHIP BETWEEN PRESIDENT AND
ADMINISTRATION DATA *

CALL SOM(PRESEN, PORDER, IERR)
INC = NUM

* PROCESS EACH OCCURRENCE OF ADMINISTRATION DATA *

DJ 17 L=1, INC

* READ ADMINISTRATION IDENTIFIER *

CALL ORSAD(10, BUF, EOF)

* TEST TO SEE IF ADMINISTRATION IS ALREADY STORED
IN DATA BASE *

CALL FMSK(AORDER, BUF, IERR)
IF (IERR .GT. -1) GOTO 16

* IF SO, GOTO 16; IF NOT, CREATE ADMINISTRATION RECORD
AND STORE IDENTIFIER IN 'IDENT' *

CALL CR(ADMIN,KEY, IERR)
CALL SFR(IDENT, ADMIN, BUF, IERR)

* ADD ADMINISTRATION RECORD TO SET 'AORDER' *

CALL A8S(AORDER, ADMIN, IERR)
GOTO 17

* MAKE CURRENT MEMBER OF SET 'AORDER', CURRENT
RECORD OF TYPE 'ADMIN' *

16 CALL SSM(AORDER, IERR)

* FORM RELATIONSHIP BETWEEN PRESIDENT AND ADMINISTRATION
RECORD *

17 CALL A8S(PRESEN, ADMIN, IERR)

* READ NUMBER OF CONGRESSES AND CONVERT TO NUMBER *
CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
INC=NUM

* PROCESS EACH OCCURRENCE OF CONGRESS DATA *

DO 18 I=1, INC

* READ CONGRESS IDENTIFIER *

CALL QREAD(10, BUF, EOF)

* TEST TO SEE IF CONGRESS IS ALREADY STORED IN DATA BASE *

CALL FMSK(CORDER, BUF, IFRR)
IF (IERR .GT. -1) GOTO 19

* IF SO, GO TO 19; IF NOT, CREATE CONGRESS RECORD AND STORE IDENTIFIER IN 'IDENT' *

CALL CR(CONGRS, KEY, IERR)
CALL SFK(IDENT, CONGRS, BUF, IERR)

* ADD CONGRESS RECORD TO SET 'CORDER' *

CALL AMS(CORDER, CONGRS, IERR)

* MAKE CURRENT MEMBER OF SET 'CORDER', CURRENT RECORD OF TYPE 'CONGRS' *

CALL SOM(CONGPS, CORDER, IERR)

* FORM RELATIONSHIP BETWEEN PRESIDENT AND CONGRESS RECORD THROUGH USE OF 'NUB' RECORD CONNECTION *

CALL SCM(PRESCS, PORDER, IERR)
CALL CR(NUB, KEY, IERR)
CALL AMS(CONGPS, NUB, IERR)
CALL AMS(PRESCS, NUB, IERR)
RETURN

* PROCESSING OF DATA RELATED TO ONE PRESIDENT IS DONE *

END
SUBROUTINE CRAN

***** THIS ROUTINE HANDLES ALL DATA FOLLOWING AN ADMINISTRATION IDENTIFIER *

* COMMON AREAS FOR CRAN *

COMMON/NAMS/PRES,IDENT,FSTNAM,INITAL,SURNAM,
& MONTH8,DAYB,YEARB,HEIGHT,PARTY,COLEG,
& ANSTRY,RELIGN,MONTHD,DAYD,YEARD,CAUSE,
& FATHER,MOTHER,NOOC,OCUP,MARRGE,WNFE,
& MONTHM,DAYM,YEARM,CHILDN,ADMIN,MONTH,
& DAY,YEAR,VPRES,STATE,NAME,YEARAD,
& CAPIAL,AREA,ARANK,POP,PRANK,VOTES,
& CITY,ELCET,WINNER,OPPON,CONGRS,SENATE,
& NUMBER,HSEREP,PRESM,PRESCS,PRESEN,
& PRESAN,NUSTAT,CITIES,CABINT,SMEMBS,HMEMBS,
& STASIZ,ORDER,ORDER,ORDER,ORDER,ORDER,
& ELECTO,PRESS,NUB,CONGPS

INTEGER PRES(2),IDENT(2),FSTNAM(2),INITAL(2),SURNAM(2),
& MONTHB(2),DAYB(2),YEARB(2),HEIGHT(2),PARTY(2),COLEG(2),
& ANSTRY(2),RELIGN(2),MONTHD(2),DAYD(2),YEARD(2),CAUSE(2),
& FATHER(2),MOTHER(2),NOOC(2),OCUP(2),MARRGE(2),WNFE(2),
& MONTHM(2),DAYM(2),YEARM(2),CHILDN(2),ADMIN(2),MONTH(2),
& DAY(2),YEAR(2),VPRES(2),STATE(2),NAME(2),YEARAD(2),
& CAPIAL(2),AREA(2),ARANK(2),POP(2),PRANK(2),VOTES(2),
& CITY(2),ELCET(2),WINNER(2),OPPON(2),CONGRS(2),SENATE(2),
& NUMBER(2),HSEREP(2),PRESM(2),PRESCS(2),PRESEN(2),
& PRESAN(2),NUSTAT(2),CITIES(2),CABINT(2),SMEMBS(2),HMEMBS(2),
& STASIZ(2),ORDER(2),ORDER(2),ORDER(2),ORDER(2),ORDER(2),
& ELECTO(2),PRESS(2),NUB(2),CONGPS(2)

COMMON/PARS/NUM,BUF,EOF

INTEGER NUM,BUF(3)

LOGICAL EOF

* READ THE ADMINISTRATION IDENTIFIER FROM PRESFILE *

CALL QREAD(10,BUF,EOF)

* TEST TO SEE IF THE ADMINISTRATION IS ALREADY IN THE DATA BASE *

CALL FMSK(AORDER,BUF,IERR)
IF (IERR .GT. -1) GOTO 30

* IF SO, GO TO 30; IF NOT, CREATE A RECORD OCCURRENCE FOR IT AND STORE THE IDENTIFIER IN 'IDENT' *

CALL CRADMIN(ADMIN,KEY,IERR)
CALL SFR(IDENT,ADMIN,BUF,IERR)

* ADD THIS RECORD TO THE SET 'AORDER' *

CALL AMS(AORDER,ADMIN,IERR)
GOTO 301

* MAKE THE RECORD THE CURRENT RECORD OCCURRENCE OF THE RECORD TYPE 'ADMIN' *
CALL SRM(AORDER,IERR)

* READ MONTH THE ADMINISTRATION STARTED AND STORE IN 'MONTH' *

CALL QREAD(10,BUF,EOF)
CALL SFRI(MONTH,ADMIN,BUF,IERR)

* READ DAY THE ADMINISTRATION STARTED, CONVERT TO NUMBER, AND STORE IN 'DAY' *

CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)
CALL SFRI(DAY,ADMIN,NUM,IERR)

* READ YEAR THE ADMINISTRATION STARTED, CONVERT TO NUMBER, AND STORE IN 'YEAR' *

CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)
CALL SFRI(YEAR,ADMIN,NUM,IERR)

* PRESIDENT NAME IS READ AND IGNORED (THIS INFORMATION IS ALSO GIVEN IN PRESIDENT DATA) *

CALL QREAD(10,BUF,EOF)

* READ NUMBER OF CABINET RECORDS AND CONVERT TO NUMBER *

CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)

* IF NO CABINETS, GO TO 31, OTHERWISE, SPECIFY RELATIONSHIP BETWEEN ADMINISTRATION AND CABINET RECORD *

IF (NUM .EQ. 0) GOTO 31
CALL SOM(CABINT,AORDER,IERR)
INC=NUM

* PROCESS EACH OCCURRENCE OF CABINET DATA *

DO 32 I=1,INC

* CREATE CABINET RECORD OCCURRENCE *

CALL CRI(VPRES,KEY,IERR)

* READ FIRST NAME OF VICE PRESIDENT AND STORE IN 'FSTNAM' *

CALL QREAD(10,BUF,EOF)
CALL SFRI(FSTNAM,VPRES,BUF,IERR)

* READ LAST NAME OF VICE PRESIDENT AND STORE IN 'SURNAM' *

CALL QREAD(10,BUF,EOF)
CALL SFRI(SURNAM,VPRES,BUF,IERR)

* FORM RELATIONSHIP BETWEEN ADMINISTRATION AND
CABINET RECORD *

CALL AMS(CABINT,VPRES,IERR)

* READ NUMBER OF STATES ADMITTED DURING ADMINISTRATION
  AND CONVERT TO NUMBER *

CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)

* IF NO STATES ADMITTED, STOP; OTHERWISE, SPECIFY
  RELATIONSHIP BETWEEN ADMINISTRATION AND STATE RECORD *

IF (NUM .EQL. 0) RETURN
CALL SOM(NUSTAT,AORDER,IERR)
INC = NUM

* PROCESS EACH OCCURRENCE OF STATE DATA *

DO 34 I=1,INC

* READ STATE NAME(AND STATE IDENTIFIER) ADMITTED DURING
  THE ADMINISTRATION. ONLY THE STATE IDENTIFIER NEED
  BE USED TO FORM THE RELATIONSHIP BETWEEN 'ADMIN'
  AND 'STATE' *

CALL QREAD(10,BUF,EOF)
CALL QREAD(10,BUF,EOF)

* TEST TO SEE IF THE STATE IS ALREADY IN THE DATA BASE *

CALL FMSK(SORDER,BUF,IERR)
IF (IERR .GT. -1) GOTO 35

* IF SO, GO TO 35; IF NOT, CREATE A RECORD OCCURRENCE
  FOR IT AND STORE STATE IDENTIFIER IN 'IDENT' *

CALL CR(STATE,KEY,IERR)
CALL SFRI(IDENT,STATE,BUF,IERR)

* ADD STATE RECORD OCCURRENCE TO SET 'SORDER' *

CALL AMS(SORDER,STATE,IERR)
GOTO 34

* MAKE THE RECORD THE CURRENT RECORD OCCURRENCE
  OF THE RECORD TYPE 'STATE' *

35 CALL SRM(SORDER,IERR)

* FORM RELATIONSHIP BETWEEN ADMINISTRATION AND
  STATE RECORD *

34 CALL AMS(NUSTAT,STATE,IERR)
RETURN

* PROCESSING OF DATA RELATED TO ONE ADMINISTRATION IS DONE *

END
**THIS ROUTINE HANDLES ALL DATA FOLLOWING A ELECTION DATA IDENTIFIER **

* COMMON AREAS FOR CRET *

**COMP/AM/SP/PRES, IDENT, FSTNM, INITAL, SURNAM, & MONTH, DAY, YEAR, HEIGHT, PARTY, COLLEGE, & ANSTRY, RELION, MONTH, CAYC, YEARC, CAUSE, & FATHER, MOTHER, NOCC, OCCUP, MARRGE, WIFE, & MOTHER, DAYM, YEARM, CHILDN, ADMIN, MOUTH, & DAY, YEAR, VPress, STATE, NAME, YEARAD, & CAPITAL, AREA, ARANK, POP, PRANK, VOTES, & CITY, ELECTION, WINNER, CPPGN, CONCRS, SENATE, & NUMBER, HSLREP, PRESPE, PPRESCS, PRESN, & PRESAN, NUSAT, CITIES, CABINT, SMEMBS, HMEMBS, & STASIZ, PORDER, SCORDER, CRORDER, RORDER, AORDER, & ELECTV, VPress, NUB, CONGPS

INTEGR PRES(2), IDENT(2), FSTNM(2), INITAL(2), SURNAM(2), & MONTH(2), DAY(2), YEAR(2), HEIGHT(2), PARTY(2), COLLEGE(2), & ANSTRY(2), RELION(2), MONTH(2), DAYM(2), YEARM(2), CAUSE(2), & FATHER(2), MOTHER(2), NOCC(2), OCCUP(2), MARRGE(2), WIFE(2), & MOTHER(2), DAYM(2), YEARM(2), CHILDN(2), ADMIN(2), MOUTH(2), & DAY(2), YEAR(2), VPress(2), STATE(2), NAME(2), YEARAD(2), & CAPITAL(2), AREA(2), ARANK(2), PCLP(2), PRANK(2), VOTES(2), & CITY(2), ELECTION(2), WINNER(2), CPPGN(2), CONCRS(2), SENATE(2), & NUMBER(2), HSLREP(2), PRESPE(2), PPRESCS(2), PRESN(2), & PRESAN(2), NUSAT(2), CITIES(2), CABINT(2), SMEMBS(2), HMEMBS(2), & STASIZ(2), PORDER(2), SCORDER(2), CRORDER(2), RORDER(2), AORDER(2), & ELECTV(2), VPress(2), NUB(2), CONGPS(2)

COMP/AM/PRES/NUM, BUF, ECF

INTEGER NUM, BUF(3)

LOGICAL ECF

**READ THE ELECTION IDENTIFIER FROM PRESFILE **

CALL QREAD(10, BUF, ECF)

**TEST TO SEE IF THE ELECTION IS ALREADY IN THE DATA BASE **

CALL FMSK(EORDER, BUF, IERR)
If (IERR .GT. -1) GOTO 20

* IF SO, GO TO 20; IF NOT, CREATE A RECORD OCCURRENCE FOR IT AND STORE THE IDENTIFIER *

CALL CI(ELECT, KEY, IERR)
CALL SP(IIDENT, ELECT, BUF, IERR)

* AND THIS RECORD OCCURRENCE TO THE SET 'EORDER' *

CALL ANS(EORDER, ELECT, IERR)
GOTO 201

* MAKE THE RECORD THE CURRENT RECORD OCCURRENCE IF THE RECORD TYPE 'ELECT' *
20 CALL SHF(ELECT, IERR)
   * READ THE YEAR OF THE ELECTION, CONVERT TO NUMBER,
     AND STORE IN 'YEAR' *
201 CALL QREAD(10, BUF, IOUT)
   CALL CTOI(BUF, 1, 10, NUM)
   CALL SHF(YEAR, ELECT, NUM, IERR)
   * READ THE WINNER OF THE ELECTION AND STORE IN 'WINNER' *
   CALL QREAD(10, BUF, IOUT)
   CALL SHF(WINNER, ELECT, BUF, IERR)
   * READ THE WINNING PARTY AND STORE IN 'PARTY' *
   CALL QREAD(10, BUF, IOUT)
   CALL SHF(PARTY, ELECT, BUF, IERR)
   * READ THE NUMBER OF WINNING VOTES, CONVERT TO NUMBER,
     AND STORE IN 'VOTES' *
   CALL QREAD(10, BUF, IOUT)
   CALL CTOI(BUF, 1, 10, NUM)
   CALL SHF(VOTES, ELECT, NUM, IERR)
   * READ NUMBER OF OPPONENTS AND CONVERT TO NUMBER *
   CALL QREAD(10, BUF, IOUT)
   CALL CTOI(BUF, 1, 10, NUM)
   * SPECIFY RELATIONSHIP BETWEEN ELECTION AND OPPONENT
     RECORD *
   CALL SHF(ELECT, ECPDIF, IERR)
   INC=NUM
   * PROCESS EACH OCCURRENCE OF OPPONENT DATA *
211 I=1, INC
   * CREATE AN OCCURRENCE OF AN OPPONENT RECORD AND
     FORM RELATIONSHIP BETWEEN OPPONENT AND ELECTION
     RECORD *
   CALL CR(OPPON, KEY, IERR)
   CALL AMS(ELECT, OPPON, IERR)
   * READ THE NAME OF OPPONENT AND STORE IN NAME *
   CALL QREAD(10, BUF, IOUT)
   CALL SHF(NAME, OPPON, BUF, IERR)
   * READ OPPONENT'S PARTY AND STORE IN 'PARTY' *
   CALL QREAD(10, BUF, IOUT)
   CALL SHF(PARTY, OPPON, BUF, IERR)
   * READ NUMBER OF OPPONENT'S VOTES, CONVERT TO NUMBER,
     AND STORE IN 'VOTES' *
   CALL QREAD(10, BUF, IOUT)
   CALL SHF(VOTES, OPPON, BUF, IERR)
CALL QREAD(10, BUF, ECF)
CALL CTOI(BUF, 1, 10, NUM)
21 CALL SFR(VOTES, OPPON, NUM, IERR)
RETURN

* PROCESSING OF DATA RELATED TO ONE ELECTION IS DONE *

END
** ** THIS ROUTINE HANDLES ALL DATA FOLLOWING A CONGRESS DATA IDENTIFIER ** **

* COMMON AREAS FOR CRCS *

COMMON/NAMPS/PRES,IDENT,FSTNM,INITAL,SURNAM, 
& MONTH8, DAYB,YEARB,HEIGHT,PARTY,COLEGE, 
& ANSTRY, RELIGN,MONTHD,DAYD,YEARD,CAUSE, 
& FATHER,MOTHER,NOOCC, OCCUP,MARRGE,WIFE, 
& MONTHM,DAYM,YEARM,CHILDM, ADMIN, MONTH, 
& DAY,YEAR,VPRES,STATE, NAME, YEARAD, 
& CAPITAL, AREA, ARANK, POP, PRANK, VOTES, 
& CITY, ELECT, WINNER, OPPON, CONGRS, SENATE, 
& NUMBER, HSEREP, PRESME, PRESCS, PRESEN, 
& PRESAN, NUSTAT, CITIES, CABINT, SMEMBS, HMEMBS, 
& STASIZ, PORDER, SORDER, CORDER, EORDER, AORDER, 
& ELECTO, PRESS, NUB, CONGPS

INTEGER PRES(2), IDENT(2), FSTNM(2), INITAL(2), SURNAM(2), 
& MONTH8(2), DAYB(2), YEARB(2), HEIGHT(2), PARTY(2), COLEGE(2), 
& ANSTRY(2), RELIGN(2), MONTHD(2), DAYD(2), YEARD(2), CAUSE(2), 
& FATHER(2), MOTHER(2), NOOCC(2), OCCUP(2), MARRGE(2), WIFE(2), 
& MONTHM(2), DAYM(2), YEARM(2), CHILDM(2), ADMIN(2), MONTH(2), 
& DAY(2), YEAR(2), VPRES(2), STATE(2), NAME(2), YEARAD(2), 
& CAPITAL(2), AREA(2), ARANK(2), POP(2), PRANK(2), VOTES(2), 
& CITY(2), ELECT(2), WINNER(2), OPPON(2), CONGRS(2), SENATE(2), 
& NUMBER(2), HSEREP(2), PRESME(2), PRESCS(2), PRESEN(2), 
& PRESAN(2), NUSTAT(2), CITIES(2), CABINT(2), SMEMBS(2), HMEMBS(2), 
& STASIZ(2), PORDER(2), SORDER(2), CORDER(2), EORDER(2), AORDER(2), 
& ELECTO(2), PRESS(2), NUB(2), CONGPS(2)

COMMON/ PARS/ NUM/ BUF/ EOF
INTEGER NUM/ BUF(3)

LOGICAL EOF

CALL QREAD(10, BUF, EOF)

* TEST TO SEE IF THE ELECTION IS ALREADY IN THE DATA BASE *

CALL FMSK(CORDER, BUF, IERR)
IF (IERR .GT. -1) GOTO 401

* IF SO, GO TO 401; IF NOT, CREATE A RECORD OCCURRENCE FOR IT AND STORE THE IDENTIFIER *

CALL CR(CONGRS, KEY, IERR)
CALL SFR(IDENT, CONGRS, BUF, IERR)

* ADD THIS RECORD OCCURRENCE TO THE SET 'CORDER' *

CALL AMS(CORDER, CONGRS, IERR)
GOTO 40

* MAKE THE RECORD THE CURRENT RECORD OCCURRENCE OF THE RECORD TYPE 'CONGRS' *
CALL SRM(CORDER, IERR)

* READ THE NUMBER OF SENATE PARTIES AND CONVERT TO NUMBER *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)

* SPECIFY RELATIONSHIP BETWEEN CONGRESS AND SENATE RECORD *

CALL SOM(SMEMBS, CORDER, IERR)
INC=NUM

* PROCESS EACH OCCURRENCE OF SENATE DATA *

DO 41 I=1, INC

* CREATE AN OCCURRENCE OF A SENATE RECORD *

CALL CR(SENATE, KEY, IERR)

* READ THE SENATE PARTY AND STORE IN 'PARTY' *

CALL QREAD(10, BUF, EOF)
CALL SFR(PARTY, SENATE, BUF, IERR)

* READ THE NUMBER OF MEMBERS, CONVERT TO NUMBER, AND STORE IN 'NUMBER' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
CALL SFR(NUMBER, SENATE, NUM, IERR)

* FORM RELATIONSHIP BETWEEN CONGRESS AND SENATE RECORD *

CALL AMS(SMEMBS, SENATE, IERR)

* READ THE NUMBER OF HOUSE PARTIES, CONVERT TO NUMBER *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)

* SPECIFY RELATIONSHIP BETWEEN CONGRESS AND HOUSE RECORD *

CALL SOM(HMEMBS, CORDER, IERR)
INC=NUM

* CREATE AN OCCURRENCE OF A HOUSE RECORD *

DO 42 I=1, INC
CALL CR(HSEREP, KEY, IERR)

* READ THE HOUSE PARTY AND STORE IN 'PARTY' *

CALL QREAD(10, BUF, EOF)
CALL SFR(PARTY, HSEREP, BUF, IERR)

* READ THE NUMBER OF MEMBERS, CONVERT TO NUMBER, AND STORE IN 'NUMBER' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF,1,LO,NUM)
CALL SFR(NUMBER,HSEREP,NUM,IERR)

* FORM RELATIONSHIP BETWEEN CONGRESS AND HOUSE RECORD *

CALL AMS(HMEMBS,HSEREP,IERR)
RETURN

* PROCESSING OF DATA RELATED TO ONE CONGRESS IS DONE *

END
SUBROUTINE CRSS

***** THIS ROUTINE HANDLES ALL DATA FOLLOWING A
STATE DATA IDENTIFIER *****

* COMMON AREAS FOR CRSS *

COMMON/NAMS/PRES, IDENT, FSTNAM, INITAL, SURNAM,
& MONTHB, DAYB, YEARB, HEIGHT, PARTY, COLEG, 
& ANSTRY, REEIGN, MONTHD, DAYD, YEARD, CAUSE, 
& FATHER, MOTHER, NOOCC, OCCUP, MARRGE, WIFE, 
& MONTHM, DAYM, YEARM, CHILDM, ADMINM, MONTHM, 
& DAY, YEAR, VPRES, STATE, NAME, YEARAD, 
& CAPITAL, AREA, ARANK, POP, PRANK, VOTES, 
& CITY, ELECT, WINNER, OPPON, CONGRS, SENATE, 
& NUMBER, HSEREP, PRESME, PRESCS, PRESEN, 
& PRESAN, NUSTAT, CITIES, CABINT, SMEMBS, HMEMBS, 
& STASIZ, PORDER, SORDER, CORDER, EORDER, AORDER, 
& ELECTO, PRESS, NUB, CONGPS

INTEGER PRES(2), IDENT(2), FSTNAM(2), INITAL(2), SURNAM(2), 
& MONTHB(2), DAYB(2), YEARB(2), HEIGHT(2), PARTY(2), COLEG(2), 
& ANSTRY(2), REEIGN(2), MONTHD(2), DAYD(2), YEARD(2), CAUSE(2), 
& FATHER(2), MOTHER(2), NOOCC(2), OCCUP(2), MARRGE(2), WIFE(2), 
& MONTHM(2), DAYM(2), YEARM(2), CHILDM(2), ADMINM, MONTHM, 
& DAY(2), YEAR(2), VPRES(2), STATE(2), NAME(2), YEARAD(2), 
& CAPITAL(2), AREA(2), ARANK(2), POP(2), PRANK(2), VOTES(2), 
& CITY(2), ELECT(2), WINNER(2), OPPON(2), CONGRS(2), SENATE(2), 
& NUMBER(2), HSEREP(2), PRESME(2), PRESCS(2), PRESEN(2), 
& PRESAN(2), NUSTAT(2), CITIES(2), CABINT(2), SMEMBS(2), HMEMBS(2), 
& STASIZ(2), PORDER(2), SORDER(2), CORDER(2), EORDER(2), AORDER(2), 
& ELECTO(2), PRESS(2), NUB(2), CONGPS(2)

COMMON/PARS/NUM, BUF, EOF
INTEGER NUM, BUF(3)
LOGICAL EOF

* READ THE STATE IDENTIFIER FROM PRESFILE *

CALL QREAD(10, BUF, EOF)

* TEST TO SEE IF THE STATE IS ALREADY IN THE
DATA BASE *

CALL FMSK(SORDER, BUF, IERR)
IF (IERR .GT. -1) GOTO 50

* IF SO, GO TO 50; IF NOT, CREATE A RECORD OCCURRENCE
FOR IT AND STORE THE IDENTIFIER *

CALL CR(STATE, KEY, IERR)
CALL SFRI(IDENT, STATE, BUF, IERR)

* ADD THIS RECORD OCCURRENCE TO THE SET 'SORDER' *

CALL AMS(SORDER, STATE, IERR)
GOTO 51

* MAKE THE RECORD THE CURRENT RECORD OCCURRENCE
OF THE RECORD TYPE 'STATE' *
CALL SRM(SORDER, IERR)

* READ THE NAME OF THE STATE AND STORE IN 'NAME' *

CALL QREAD(10, BUF, EOF)
CALL SFR(NAME, STATE, BUF, IERR)

* READ THE YEAR ADMITTED TO THE UNION, CONVERT TO NUMBER, 
AND STORE IN 'YEARAD' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
CALL SFR(YEARAD, STATE, NUM, IERR)

* READ THE CAPITAL CITY AND STORE IN 'CAPITAL' *

CALL QREAD(10, BUF, EOF)
CALL SFR(CAPITAL, STATE, BUF, IERR)

* READ THE AREA OF THE STATE, CONVERT TO NUMBER, 
AND STORE IN 'AREA' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
CALL SFR(STATE, AREA, NUM, IERR)

* READ THE RANK OF THE STATE (IN TERMS OF AREA SIZE), 
CONVERT TO NUMBER, AND STORE IN 'ARANK' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
CALL SFR(STATE, AREA, ARANK, IERR)

* READ THE POPULATION OF THE STATE, CONVERT TO NUMBER, 
AND STORE IN 'POP' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
CALL SFR(STATE, POP, NUM, IERR)

* READ THE RANK OF THE STATE (IN TERMS OF POPULATION SIZE), 
CONVERT TO NUMBER, AND STORE IN 'PRANK' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
CALL SFR(PRANK, STATE, NUM, IERR)

* ADD THIS RECORD OCCURRENCE TO THE SET 'STASIZ' *)

CALL AMS(STASIZ, STATE, IERR)

* READ THE NUMBER OF ELECTORAL VOTES FOR THE STATE, 
CONVERT TO NUMBER, AND STORE IN 'VOTES' *

CALL QREAD(10, BUF, EOF)
CALL CTOI(BUF, 1, 10, NUM)
CALL SFR(VOTES, STATE, NUM, IERR)

* READ NUMBER OF MAJOR CITIES AND CONVERT TO NUMBER *
CALL QREAD(10,BUF,EOF)
CALL CTOI(BUF,1,10,NUM)

   * IF NO MAJOR CITIES, STOP; OTHERWISE SPECIFY
   RELATIONSHIP BETWEEN STATE AND CITY RECORD *

IF (NUM .EQ. 0) RETURN
CALL SOMCITIES,SORDER,IERR)
INC=NUM

   * PROCESS EACH OCCURRENCE OF CITY DATA *

DO 53 I=1,INC
   * CREATE AN OCCURRENCE OF A CITY RECORD *
   CALL CR(CITY,KEY,IERR)
   * READ NAME OF CITY AND STORE IN 'NAME' *
   CALL QREAD(10,BUF,EOF)
   CALL SFR(NAME,CITY,BUF,IERR)
   * READ POPULATION OF CITY AND STORE IN 'POP' *
   CALL QREAD(10,BUF,EOF)
   CALL CTOI(BUF,1,10,NUM)
   CALL SFR(POP,CITY,NUM,IERR)
   * FORM RELATIONSHIP BETWEEN STATE AND CITY RECORD *
   CALL AMSCITIES,CITY,IERR)
RETURN

   * PROCESSING OF DATA RELATED TO ONE STATE IS DONE *
END
7. GENERATING REPORTS FROM THE DATA BASE

Once data has been successfully entered into the data base there are several routines (described in ISDOS Working Paper No. 88 and the PSA software documentation package) which can be used to aid in the retrieval process and in formatting the retrieved data.

7.1 Retrieving Data

There are several routines available to the user (of which just a few are described here) to aid in the retrieval of data in a random fashion or via some ordered manner.

- **FNM** Used to locate member record occurrences for a given owner record occurrence.
- **FFM**
- **SOM** Used to retrieve data occurrences based on some defined relationship.
- **SMM**
- **GFM** Used to retrieve data from a particular "item" occurrence.
- **GFO**
- **SMOVE** Used to retrieve data from a repeating item.

7.2 **FFM Find First Member record occurrence**

This routine is used to find the first member record occurrence for the current owner record in a given set. For example, if a listing of all President names were to be retrieved and printed the call

```
CALL FFM (PORDER, IERR)
```

finds the first member record occurrence of the set PORDER. (Looking back, notice that the DDL designated the owner record for this set to be SYSTEM and member record occurrences to be of the type PRES.) At this point the first member occurrence has been located or an error code has been returned in IERR. Data can now be taken from the record occurrence (if no error was encountered).

7.3 **FNM Find Next Member record occurrence**

This routine can be used to find the first member record occurrence (like FFM) if there is no current member record occurrence, or can be used to find any subsequent member record occurrences in the order specified in the DDL. For example,

```
CALL FNM (PORDER, IERR)
```

will find the next PRES record occurrence, based on the value of
the IDENT item. (Remember that the set PORDER is ordered on the IDENT items within the PRES record occurrences.) A return code value of -1 will be given to IERR should an end-of-file condition be encountered.

7.4 SOM Set the current Owner of a set, based on the current Member of a set.

This routine is used (as in SMM) to locate and retrieve data based on specific relationships between the record types in the data base. For example, for a PRES record occurrence (found by FFM or FN) it may be desirable to find those ELECT record occurrences related to it via the PRESEN relationship.

CALL SOM (PRESEN, PORDER, IERR)

specifies that the current member record of set PORDER (which would be a PRES record type) is to be made the current owner record of the set PRESEN (whose members are ELECT record types). Once this has been set, the first member record occurrence of the set PRESEN is available for any retrieval procedures. The next ELECT member record occurrence for the particular PRES record occurrence can be located by the FN routine:

CALL FN (PRESEN, IERR)

7.5 SMM Set the current Member of a set, based on the current Member of a set.

This routine can be used to retrieve data for a record occurrence which is a member of two different types of sets. For example, all PRES record occurrences are members of the set PORDER (ordering of all Presidents by identifier) as well as the set PRESS (relationship between Presidents and State born in). So, given that a particular PRES record occurrence is available via FFM or FN, the STATE record occurrence the President is related to can be found by the call

CALL SMM (PRESS, PORDER, IERR)

where the current member of the set PORDER also now becomes the current member of the set PRESS. State data can now be extracted.

7.6 GFM Get Field from Member record occurrence

Once the appropriate record occurrence has been found by the above routines, GFM is the routine used to actually retrieve the data from the record. For example, to retrieve the first name of a President the call

CALL GFM (FSTNAM, PORDER, BUF, IERR)

would be used. This retrieves the data from the "item" FSTNAM in the current member record occurrence of the set PORDER (which means this is a PRES record occurrence) and places the data in BUF. BUF must be of the same data type (i.e. integer, character, decimal, etc.) as the "item" and large enough to hold its value.
7.7 GFO Get Field from Owner record occurrence

This routine is basically the same as GFM, but retrieves data from an owner record occurrence rather than a member. Looking back at the SMM routine, a STATE record occurrence was found as a result of

CALL SMM (PRESS, PORDER, IERR)

which also made the STATE record occurrence the current owner of the set PRESS. GFO allows data from the STATE record to be retrieved.

CALL GFO (NAME, PRESS, BUF, IERR)

retrieves the data in the "item" NAME from the current owner record occurrence of the set PRESS, and puts the data into BUF.

7.8 Report Routines

As the data is being retrieved it has to be formatted in some manner to be presented as a report. The following routines aid in this process.

HEDING Used to print a heading for the report

BUFBLD }
NINBUF } Used to store data in the output buffer to be printed

PBUF Used to print out contents of the output buffer

7.9 HEDING (Heading)

This subroutine allows the programmer to specify a title for a report. If the report is more than one page long, the title will be presented at the top of each new page. For example, by specifying

CALL HEDING(31,31H"** PRESIDENT/ELECTION REPORT **",0,3)

the title "** PRESIDENT/ELECTION REPORT **" will be printed at the top of each page of the report. The first parameter value, 31, designates that the title is 31 characters long. The zero (0) parameter value above specifies that the title should be centered. (Any positive integers in place of this parameter value specifies the column where the title begins.) Finally, the last parameter value, 3, designates the number of spaces to skip before printing out the contents of the report.

7.10 BUFBLD (Buffer Build)

This routine allows the storage of character data into an output buffer when printed, becomes part of the report. For example,

CALL BUFBLD(1,1,14,14HPRESIDENT NAME)

stores the character string "PRESIDENT NAME" into the first
position (1) in the output buffer (as designated by the first parameter). The third parameter (here 14) specifies the number of characters in the string. Taking another example,

```plaintext
CALL BUFBLD(28,1,3,BUF)
```

this specifies that the contents of the variable (a character array) BUF, should be stored in the output buffer, starting at the 28th position in the buffer. The second parameter specifies an index into BUF where the character string starts. (In this case the character data of interest starts in the first (1) position.) 3 specifies the length of the character string to be stored. Note that if BUF contained a twelve character string, the last 9 characters would be ignored while length is specified to be 3.

### 7.11 NINBUF (Numeric Integer into Buffer)

This routine is used much in the same way as BUFBLD but, instead places numeric data (rather than character) into the output buffer. For example,

```plaintext
CALL NINBUF(NUM,21,4)
```

converts the numeric data in the variable NUM into character format and stores this starting at position 21 in the output buffer. The last parameter specifies the length of the number being stored. The number is assumed to be an integer.

### 7.12 PBUF (Print Buffer)

This routine prints out the contents of the output buffer set up by the BUFBLD and NINBUF routines. By specifying

```plaintext
PBUF(0,1,.TRUE.)
```

The output buffer is printed, a line is skipped after the line is printed (1), and the buffer is blanked out (.TRUE.). The zero (0) parameter designates that no lines should be skipped before printing the buffer. (Of course, this can have any positive integer value.) If the buffer is not blanked out (.FALSE.) data stored in the buffer will be overlayed over the previous data.

### 7.13 Code to Generate Reports from the Presidential Data Base

Now that the basic routines needed to generate reports from a data base have been described, examples of how these routines can be used to actually generate reports will be given in this section.

The common areas and block data used in the population of the data base can be used in these report programs to reduce effort in defining variables and constants.
Assuming the source code for the report program to be in TESTPROG, the object code can be put into TEST when

$RUN  *FTN  PAR=SOURCE=TESTPROG  LOAD=TEST

is given. Assuming BLOCKFILE contains the object version of the block data subprogram for the data base the report program can be executed by the command:

$RUN  TEST+BLOCKFILE+SELW:DBLIB  2=DBF  3=DBTF

where DBF is the data base file and DBTF is the data base table file.

The remainder of this section presents descriptions and listings of programs that generate reports from the Presidential data base. Appendix B presents the reports generated from these programs.
TEST PROGRAM #1

This program obtains an alphabetical listing of all presidents in the data base. It prints out first and last name and middle initial if there is one. For each president, it also finds all elections related to that president that won him the presidency and the date of the election, and at least one of the opponents he beat. In some cases a particular opponent is not specified but rather that a number of opponents were defeated. For those presidents with no election information, nothing is given.
**** PRESIDENT/ELECTION REPORT PROGRAM ****

* THIS PROGRAM RETRIEVES AND PRINTS ELECTION INFORMATION RELATIVE TO EACH PRESIDENT *

* COMMON AREAS FOR THE PROGRAM *

COMMON/NAMS/PRES,IDENT,FSTNAME,INITIAL,SURNAM,
& MONTHB,DAYE,YEARB,HEIGHT, PARTY, COLEGE,
& ANSTRY,RELG,MCNTHD,DAYD,YEARD, CAUSE,
& FATHER,MOTHER,NOC,CUP,MARRGE,WIFE,
& MONTHM,DAYM,YEARM,CHILD,MADMIN,MONTH,
& DAY,YEAR,VPRES,STATE,NAME,YEARAD,
& CAPITAL,AREA,ARRANK,POP,PRANK,VOTES,
& CITY, ELECT, WINNER, OPPON, CONGRS, SENATE,
& NUMBER, HSEREP, PRESH, PRESCS, PRESEN,
& PRESAN, NUSTAT, CITIES, CABINT, SMBBS, HMEMBS,
& STASIZ, PORDER, SORDER, CORDER, EORDER, AORDER,
& ELECTO, PRESS, NUB, CONGPS

INTEGER PRES(2), IDENT(2), FSTNAME(2), INITIAL(2), SURNAM(2),
& MONTHB(2), DAYE(2), YEARB(2), HEIGHT(2), PARTY(2), COLEGE(2),
& ANSTRY(2), RELG(2), MCNTHD(2), DAYD(2), YEARD(2), CAUSE(2),
& FATHER(2), MOTHER(2), NOCC(2), CUP(2), MARRGE(2), WIFE(2),
& MONTHM(2), DAYM(2), YEARM(2), CHILD(2), MADMIN(2), MONTH(2),
& DAY(2), YEAR(2), VPRES(2), STATE(2), NAME(2), YEARAD(2),
& CAPITAL(2), AREA(2), ARANK(2), POP(2), PRANK(2), VOTES(2),
& CITY(2), ELECT(2), WINNER(2), OPPON(2), CONGRS(2), SENATE(2),
& NUMBER(2), HSEREP(2), PRESH(2), PRESCS(2), PRESEN(2),
& PRESAN(2), NUSTAT(2), CITIES(2), CABINT(2), SMBBS(2), HMEMBS(2),
& STASIZ(2), PORDER(2), SORDER(2), CORDER(2), EORDER(2), AORDER(2),
& ELECTO(2), PRESS(2), NUB(2), CONGPS(2)

COMMON/PARS/NUM, BUF, EOF
INTEGER NUM, BUF(3)
LOGICAL EOF

* REPORT INITIALIZATION PROCEDURES *

CALL REPRINT
CALL NEWREP(*.FALSE.)

* SPECIFY THE TITLE TO BE PRINTED AT TOP OF EACH PAGE OF THE REPORT *

CALL HEDING(31,31H** PRESIDENT/ELECTION REPORT **,O,3)

* OPEN THE DATA BASE FILE AND THE DATA BASE TABLE FILE (LOGICAL I/O UNITS 2 AND 3 RESPECTIVELY) *

CALL OPEN(2,3,100,IERR)

* FIND PRESIDENT RECORD CCCURRENCE. IF NONE CAN BE FOUND, STOP *

CALL FNM(PORDER,IERR)
IF (IERR .EQ. -1) GOTO 100

* PLACE COMMENT INTO OUTPUT BUFFER *

CALL BUFBLD(1,1,14,14HPRESIDENT NAME)
* RETRIEVE PRESIDENT S FIRST NAME, MIDDLE INITIAL AND
SURNAME AND PLACE THEM INTO THE OUTPUT BUFFER *

CALL GFM(FSTNAM, PORCER, BUF, IERR)
CALL BUFBLD(16, 1, 10, BUF)
CALL GFM(INITIAL, PORCER, BUF, IERR)
CALL BUFBLD(28, 1, 3, BUF)
CALL GFM(SURNAM, PORCER, BUF, IERR)
CALL BUFBLD(31, 1, 10, BUF)

* PRINT OUT BUFFER WITH PRESIDENT S NAME *

CALL PBUF(2, 0, .TRUE.)

* TEST TO SEE IF THE PRESIDENT WON ANY ELECTIONS FOR THE
PRESIDENCY AND IF NOT, GO PROCESS THE NEXT PRESIDENT *

CALL SOM(PRESEN, PORCER, IERR)
5 IF (IERR .EQ. -1) GOTO 1

* PLACE COMMENT INTO OUTPUT BUFFER *

CALL BUFBLD(4, 1, 29, 29HWIN ELECTION OF AGAINST)

* RETRIEVE YEAR PRESIDENT WON ELECTION AND PLACE IN OUTPUT
BUFFER *

CALL GFM(YEAR, PRESEN, NUM, IERR)
CALL NINBUF(NUM, 21, 4)

* FIND OPPONENT PRESIDENT BEAT IN THIS ELECTION AND PLACE
INTO OUTPUT BUFFER *

CALL SOM(ELECTO, PRESEN, IERR)
CALL GFM(NAME, ELECTC, BUF, IERR)
CALL BUFBLD(34, 1, 10, BUF)

* PRINT OUT OUTPUT BUFFER WITH ELECTION INFORMATION *

CALL PBUF(0, 0, .TRUE.)

* CONTINUE TO PROCESS ANY REMAINING ELECTION RECORDS
RELATED TO PRESIDENT *

CALL FNM(PRESEN, IERR)
GOTO 25

* CLOSE THE DATA BASE *

100 CALL CLOS(0, ARRAY, IERR)
END
TEST PROGRAM #2

This program obtains an alphabetical listing of all states in the data base. For each state it prints out state name and year admitted to the union. It also finds any city records related to that state and prints out city name as well as city population.
**** STATE/CITY REPORT PROGRAM ****

* THIS PROGRAM RETRIEVES CITY INFORMATION AND HISTORICAL INFORMATION RELATIVE TO EACH STATE *

* COMMON AREAS FOR THE PROGRAM *

COMMON/NAMS/PRES, IDENT, FSTNAM, INITAL, SURNAM, & MONTHS, DAYB, YEARB, HEIGHT, PARTY, COLLEGE, & ANSTRY, RELIGN, MOUTH, DAYD, YEARD, CAUSE, & FATHER, MOTHER, NOOCC, OCCUP, MARRGE, WIFE, & MOUTH, DAYM, YEARM, CHILDN, ADMIN, MONTH, & DAY, YEAR, VPRES, STATE, NAME, YEARD, & CAPITAL, AREA, ARANK, POP, PRANK, VOTES, & CITY, ELECT, WINNER, OPPON, CONGRS, SENATE, & NUMBER, HSEREP, PRESME, PRESCS, PRESEN, & PRESAN, NUSTAT, CITIES, CABINT, SEMMBS, HMEMBS, & STASIZ, PORDER, SORDER, ORDER, AORDER, & ELECTO, PRESS, NUB, CONGPS

INTEGER PRES(2), IDENT(2), FSTNAM(2), INITAL(2), SURNAM(2), & MONTHS(2), DAYB(2), YEARB(2), HEIGHT(2), PARTY(2), COLLEGE(2), & ANSTRY(2), RELIGN(2), MOUTH(2), DAYD(2), YEARD(2), CAUSE(2), & FATHER(2), MOTHER(2), NOOCC(2), OCCUP(2), MARRGE(2), WIFE(2), & MOUTH(2), DAYM(2), YEARM(2), CHILDN(2), ADMIN(2), MONTH(2), & DAY(2), YEAR(2), VPRES(2), STATE(2), NAME(2), YEARD(2), & CAPITAL(2), AREA(2), ARANK(2), POP(2), PRANK(2), VOTES(2), & CITY(2), ELECT(2), WINNER(2), OPPON(2), CONGRS(2), SENATE(2), & NUMBER(2), HSEREP(2), PRESME(2), PRESCS(2), PRESEN(2), & PRESAN(2), NUSTAT(2), CITIES(2), CABINT(2), SEMMBS(2), HMEMBS(2), & STASIZ(2), PORDER(2), SORDER(2), ORDER(2), AORDER(2), & ELECTO(2), PRESS(2), NUB(2), CONGPS(2)

COMMON/PARS/NUM, BUF, EOF

INTEGER NUM, BUF(3)

LOGICAL EOF

* REPORT INITIALIZATION PROCEDURES *

CALL REPINT
CALL NEWREP(.FALSE.)

* SPECIFY THE TITLE TO BE PRINTED AT THE TOP OF EACH PAGE OF THE REPORT *

CALL HEDING(27, 27H** STATE AND CITY REPORT **, 0, 3)

* OPEN THE DATA BASE FILE AND DATA BASE TABLE FILE 1 LOGICAL I/O UNITS 2 AND 3 RESPECTIVELY *

CALL OPEN(2, 3, 100, IERR)

* FIND STATE RECORD OCCURRENCE. IF NONE CAN BE FOUND, STOP *

3 CALL FNM(SORDER, IERR)
  IF (IERR .EQ. -1) GOTO 2

* PLACE COMMENT INTO OUTPUT BUFFER *

CALL BUFBLD(1, 11, 11HSTATE NAME—)
* RETRIEVE STATE NAME, PLACE IN BUFFER, AND PRINT OUT *

CALL GFM(NAME,SORDER,BUF,IERR)
CALL BUFBLD(13,1,10,BUF)
CALL PBUF(2,0,.TRUE.)

* PLACE COMMENT INTO OUTPUT BUFFER *

CALL BUFBLD(4,1,20,20HADMITTED IN THE YEAR)

* RETRIEVE YEAR STATE WAS ADMITTED TO UNION, PLACE IN BUFFER, AND PRINT OUT *

CALL GFM(YEARAD,SORDER,NUM,IERR)
CALL NINBUF(NUM,26,4)
CALL PBUF(0,0,.TRUE.)

* TEST TO SEE IF THERE ARE ANY CITY RECORDS RELATED TO THIS STATE. IF NOT, GO PROCESS NEXT STATE RECORD *

CALL SOM(CITIES,SORDER,IERR)
IF (IERR .EQ. -1) GOTO 3

* PLACE COMMENT INTO OUTPUT BUFFER AND PRINT OUT *

CALL BUFBLD(4,1,32,32HMAJOR CITIES ARE POPULATION)
CALL PBUF(0,0,.TRUE.)

* RETRIEVE CITY NAME AND CITY POPULATION AND PUT INTO OUTPUT BUFFER *

CALL GFM(NAME,CITIES,BUF,IERR)
CALL BUFBLD(4,1,10,BUF)
CALL GFM(POP,CITIES,NUM,IERR)
CALL NINBUF(NUM,23,10)

* PRINT OUT CITY NAME AND CITY POPULATION *

CALL PBUF(0,0,.TRUE.)

* FIND NEXT CITY RECORD RELATED TO STATE. IF NONE, PROCESS NEXT STATE RECORD *

CALL FNM(CITIES,IERR)
IF (IERR .EQ. -1) GOTO 3
GOTO 1

* CLOSE THE DATA BASE *

CALL CLOS(0,ARRAY,IERR)
END
TEST PROGRAM #3

This program gets an alphabetical listing of presidents. For each president it also finds out in what year and to whom the president was married by retrieving all occurrences of the marriage records associated to the president. It also finds all administration records associated to that president and for each administration record, the vice president related to that administration. If that isn't enough, the state the president was born in is also found and the state name is printed out.
**** PRESIDENTIAL INFORMATION REPORT PROGRAM ****

* THIS PROGRAM RETRIEVES INFORMATION ABOUT EACH PRESIDENT 
  SUCH AS MARITAL DATE, WIFE'S NAME, YEARS OF ADMINISTRATION, 
  VICE PRESIDENTS AND STATE OF BIRTH *

* COMMON AREAS FOR THE PROGRAM *

COMMON/NAMS/PRES,IDENT,FSTNAM,INITAL,SURNAM, 
& MONTHB,DAYB,YEARB,HEIGHT,PARTY,COLEGE, 
& ANSTY,RELIGN,MONTHD,DAYD,YEARD,CAUSE, 
& FATHER,MOTHER,NOOCC,OCUP,MARRGE,ELADE, 
& MONTM,DAYM,YEARM,CHILDN,ADMN,MONTH, 
& DAY,YEAR,VPRES,STATE,NAME,YEARAD, 
& CAPTAL,AREA,ARANK,POP,PRANK,VOTES, 
& CITY,ELECT,WINNER,OPPON,CONGRS,SENATE, 
& NUMBER,HSEREP,PRESME,PRESCS,PRESEN, 
& PRESAN,NUSTAT,CITIES,ABINT,SMEMBS,HMEMBS, 
& STASIZ,PORDER,SORDER,CORDER,EORDER,AORDER, 
& ELECT,O PRESS,NUB,CONGPS 
INTEGER PRES(2),IDENT(2),FSTNAM(2),INITAL(2),SURNAM(2), 
& MONTHB(2),DAYB(2),YEARB(2),HEIGHT(2),PARTY(2),COREGE(2), 
& ANSTY(2),RELIGN(2),MONTHD(2),DAYD(2),YEARD(2),CAUSE(2), 
& FATHER(2),MOTHER(2),NOOCC(2),OCUP(2),MARRGE(2),ELADE(2), 
& MONTM(2),DAYM(2),YEARM(2),CHILDN(2),ADMN(2),MONTH(2), 
& DAY(2),YEAR(2),VPRES(2),STATE(2),NAME(2),YEARAD(2), 
& CAPTAL(2),AREA(2),ARANK(2),POP(2),PRANK(2),VOTES(2), 
& CITY(2),ELECT(2),WINNER(2),OPPON(2),CONGRS(2),SENATE(2), 
& NUMBER(2),HSEREP(2),PRESME(2),PRESCS(2),PRESEN(2), 
& PRESAN(2),NUSTAT(2),CITIES(2),ABINT(2),SMEMBS(2),HMEMBS(2), 
& STASIZ(2),PORDER(2),SORDER(2),CORDER(2),EORDER(2),AORDER(2), 
& ELECT(2),PRESS(2),NUB(2),CONGPS(2) 
COMMON/PARS/NUM,BUF,EOF 
INTEGER NUM,BUF(3) 
LOGICAL EOF

* REPORT INITIALIZATION PROCEDURES *

CALL REPINT 
CALL NEWREP( .FALSE. )

* SPECIFY TITLE TO BE PRINTED AT THE TOP OF EACH PAGE OF THE REPORT *

CALL HEDING(36,36H**PRESIDENTIAL INFORMATION REPORT **,.0,3) 

* OPEN THE DATA BASE FILE AND THE DATA BASE TABLE FILE 
  ( LOGICAL 1/O UNITS 2 AND 3 RESPECTIVELY ) *

CALL OPEN(2,3,100,IERR) 

* FIND A PRESIDENT RECORD OCCURRENCE. IF NONE FOUND, STOP *

CALL FM(PORDER,IERR) 
IF ( IERR .EQ. -1 ) GOTO 100 

* PLACE COMMENT INTO OUTPUT BUFFER *


IF (IIERR .EQ. -1) GOTO 30

* RETRIEVE VICE PRESIDENT'S LAST NAME AND STORE IN BUFFER *

CALL GFM (SURNAME, CABINT, BUF, IERR)
CALL BUFBLD (31, 1, 10, BUF)

* PRINT OUT BUFFER WITH ADMINISTRATION DATA *

CALL PBUF (0, 0, .TRUE.)

* TEST TO SEE IF THE PRESIDENT HAS ANY OTHER ADMINISTRATION RECORDS RELATED TO HIM. IF SO, PROCESS EACH RECORD AS ABOVE *

CALL FNM (PRESAN, IERR)
IF (IIERR .GT. -1) GOTO 3

* FIND STATE PRESIDENT WAS BORN IN, RETRIEVE NAME OF STATE AND STORE IN BUFFER WITH COMMENT *

CALL SMM (PRESS, PORDER, IERR)
CALL GFO (NAME, PRESS, BUF, IERR)
CALL BUFBLD (11, 1, 8, 8, BORN IN:)
CALL BUFBLD (10, 1, 10, BUF)

* PRINT OUT BUFFER WITH STATE INFORMATION *

CALL PBUF (0, 0, .TRUE.)

* PROCESS NEXT PRESIDENT RECORD *

GO TO 1

* CLOSE THE DATA BASE *

100 CALL CLS (N, ARRAY, IERR)
END
CALL BUFBLD(1,1,10,10HPRESIDENT)

* RETRIEVE PRESIDENT'S LAST NAME, PUT IN OUTPUT BUFFER,
AND PRINT OUT BUFFER *

CALL GFM(SURNAM,PORDER,BUF,IERR)
CALL BUFBLD(12,1,10,BUF)
CALL PBUF(2,0,TRUE.)

* FIND A MARRIAGE RECORD RELATED TO THE PRESIDENT *

CALL SOM(PRESME,PORDER,IERR)

* IF YEAR OF MARRIAGE EQUALS ZERO, A MESSAGE SHOULD BE
PRINTED THAT THE PRESIDENT WAS NOT MARRIED *

2 CALL GFM(YEAR,M,PRESME,NUM,IERR)
IF (NUM .EQ. 0) GOTO 5

* PUT COMMENT, YEAR OF MARRIAGE, AND WIFE'S NAME INTO
OUTPUT BUFFER AND PRINT OUT *

CALL BUFBLD(1,1,22,22HWAS MARRIED IN  TO)
CALL NINBUF(NUM,16,4)
CALL GFM(WIFE,PRESME,BUF,IERR)
CALL BUFBLD(24,1,10,BUF)
CALL PBUF(0,0,TRUE.)

* TEST TO SEE IF THERE ARE ANY MORE MARRIAGE RECORDS RELATED
TO THE PRESIDENT. IF SO, PROCESS EACH RECORD AS ABOVE *

CALL FName(PRESME,IERR)
IF (IERR .GT. -1) GOTO 2
GOTO 25

* PRINT MESSAGE THAT PRESIDENT WAS NOT MARRIED *

5 CALL BUFBLD(1,1,15,15HWAS NOT MARRIED)
CALL PBUF(0,0,TRUE.)

* PUT COMMENTS INTO OUTPUT BUFFER AND PRINT OUT *

25 CALL BUFBLD(1,1,25,25HHEADED ADMINISTRATIONS )
CALL BUFBLD(28,1,14,14HVICE PRESIDENT)
CALL PBUF(0,0,TRUE.)

* FIND AN ADMINISTRATION RECORD RELATED TO THE PRESIDENT *

CALL SOM(PRESAN,PORDER,IERR)

* RETRIEVE YEAR OF ADMINISTRATION AND PUT IN BUFFER *

3 CALL GFM(YEAR,PRESAN,NUM,IERR)

* TEST TO SEE IF THE PRESIDENT HAD A VICE PRESIDENT DURING
THE ADMINISTRATION. IF NOT, GO TO 30 *

CALL BUFBLD(15,1,2,2HIN)
CALL NINBUF(NUM,19,4)
CALL SOM(CABINT,PRESAN,IERR)
APPENDIX A

Presidential File (PRESFILE) Listing
<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Age</th>
<th>Height</th>
<th>Party</th>
<th>Occupation</th>
<th>State</th>
<th>Birth Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Washington George</td>
<td>February 22</td>
<td>1732</td>
<td>Virginia</td>
<td>Federalist</td>
<td>Surveyor, Farmer, Soldier</td>
<td>December 6</td>
<td>1759</td>
</tr>
<tr>
<td>John Adams</td>
<td>April 13</td>
<td>1743</td>
<td>Virginia</td>
<td>Federalist</td>
<td>Lawyer, Writer</td>
<td>July 4</td>
<td>1826</td>
</tr>
<tr>
<td>Thomas Jefferson</td>
<td>June 1</td>
<td>1772</td>
<td>Virginia</td>
<td>Federalist</td>
<td>Lawyer, Secretary</td>
<td>February 16</td>
<td>1786</td>
</tr>
<tr>
<td>James Madison</td>
<td>March 16</td>
<td>1751</td>
<td>Virginia</td>
<td>Federalist</td>
<td>Lawyer, Secretary</td>
<td>June 28</td>
<td>1836</td>
</tr>
<tr>
<td>James Ewell</td>
<td>April 28</td>
<td>1758</td>
<td>Virginia</td>
<td>Federalist</td>
<td>Lawyer, Soldier</td>
<td>July 4</td>
<td>1845</td>
</tr>
<tr>
<td>George Washington</td>
<td>February 22</td>
<td>1732</td>
<td>Virginia</td>
<td>Federalist</td>
<td>Surveyor, Farmer, Soldier</td>
<td>December 6</td>
<td>1759</td>
</tr>
<tr>
<td>Thomas Jefferson</td>
<td>April 13</td>
<td>1743</td>
<td>Virginia</td>
<td>Federalist</td>
<td>Lawyer, Writer</td>
<td>July 4</td>
<td>1826</td>
</tr>
<tr>
<td>James Madison</td>
<td>March 16</td>
<td>1751</td>
<td>Virginia</td>
<td>Federalist</td>
<td>Lawyer, Secretary</td>
<td>June 28</td>
<td>1836</td>
</tr>
<tr>
<td>James Ewell</td>
<td>April 28</td>
<td>1758</td>
<td>Virginia</td>
<td>Federalist</td>
<td>Lawyer, Soldier</td>
<td>July 4</td>
<td>1845</td>
</tr>
<tr>
<td>Month</td>
<td>Name</td>
<td>Profession</td>
<td>Date</td>
<td>Residence</td>
<td>Height</td>
<td>Party</td>
<td>Religion</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------</td>
<td>------------</td>
<td>--------</td>
<td>------------------</td>
<td>----------</td>
</tr>
<tr>
<td>January</td>
<td>Fillmore</td>
<td>Lawyer</td>
<td>1800</td>
<td>New York</td>
<td>-</td>
<td>Unitarian</td>
<td>English</td>
</tr>
<tr>
<td>February</td>
<td>Woolcarder</td>
<td>Lawyer</td>
<td>1826</td>
<td>New York</td>
<td>5' 9&quot;</td>
<td>Whig</td>
<td></td>
</tr>
<tr>
<td>November</td>
<td>Buchanan</td>
<td>Lawyer</td>
<td>1840</td>
<td>New Hampshire</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>January</td>
<td>Abraham</td>
<td>English</td>
<td>1868</td>
<td>New York</td>
<td>6' 0&quot;</td>
<td>Democratic-Dickinson</td>
<td>Scot-Irish Presbyterian</td>
</tr>
<tr>
<td>February</td>
<td>Nancy</td>
<td>English</td>
<td>1880</td>
<td>New York</td>
<td>5' 8&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>April</td>
<td>Andrew</td>
<td>English</td>
<td>1882</td>
<td>New York</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>May</td>
<td>Mary</td>
<td>English</td>
<td>1884</td>
<td>New York</td>
<td>5' 10&quot;</td>
<td>Republican-Kentucky</td>
<td>Methodist</td>
</tr>
<tr>
<td>July</td>
<td>Garfield</td>
<td>Republicana</td>
<td>1886</td>
<td>New York</td>
<td>5' 5&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>October</td>
<td>Ellen</td>
<td>Teacher</td>
<td>1888</td>
<td>New Jersey</td>
<td>5' 11&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>March</td>
<td>Grover</td>
<td>Teacher</td>
<td>1889</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>April</td>
<td>Richard</td>
<td>Teacher</td>
<td>1890</td>
<td>New Jersey</td>
<td>6' 2&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>May</td>
<td>Harrison</td>
<td>Teacher</td>
<td>1891</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>June</td>
<td>Frances</td>
<td>2 Clerk</td>
<td>1892</td>
<td>New Jersey</td>
<td>5' 6&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>July</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1893</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>August</td>
<td>Chester</td>
<td>2 Lawyer</td>
<td>1894</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>September</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1895</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>October</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1896</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>November</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1897</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>December</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1898</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>January</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1899</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>February</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1900</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>March</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1901</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>April</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1902</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>May</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1903</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>June</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1904</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>July</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1905</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>August</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1906</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>September</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1907</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>October</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1908</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>November</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1909</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>December</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1910</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>January</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1911</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>February</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1912</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>March</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1913</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>April</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1914</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>May</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1915</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>June</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1916</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>July</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1917</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>August</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1918</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>September</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1919</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>October</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1920</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>November</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1921</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>December</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1922</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>January</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1923</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>February</td>
<td>William</td>
<td>2 Lawyer</td>
<td>1924</td>
<td>New Jersey</td>
<td>5' 10&quot;</td>
<td>Republican</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Birth Year</td>
<td>Death Year</td>
<td>Occupation</td>
<td>Party</td>
<td>State</td>
<td>College</td>
<td>Nationality</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>-------</td>
<td>-------------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Theodore Roosevelt</td>
<td>1856</td>
<td>1919</td>
<td>Lawyer</td>
<td>Republican</td>
<td>New York</td>
<td>Harvard</td>
<td>Dutch</td>
</tr>
<tr>
<td>Martha 1919 RHEUMATISM</td>
<td>1900</td>
<td>1930</td>
<td>Lawyer</td>
<td>Republican</td>
<td>Ohio</td>
<td>Princeton</td>
<td>English</td>
</tr>
<tr>
<td>Louise 1863</td>
<td>1930</td>
<td>1910</td>
<td>Teacher</td>
<td>Democratic</td>
<td>Virginia</td>
<td>Harvard</td>
<td>Alphonso</td>
</tr>
<tr>
<td>William H. Taft</td>
<td>1858</td>
<td>1930</td>
<td>Lawyer</td>
<td>Republican</td>
<td>Virginia</td>
<td>Yale</td>
<td>Democratic</td>
</tr>
<tr>
<td>George Phoebe</td>
<td>1867</td>
<td>1922</td>
<td>Journalist</td>
<td>Republican</td>
<td>Vermont</td>
<td>Brown</td>
<td>Calvin</td>
</tr>
<tr>
<td>HENRY HARDING</td>
<td>1868</td>
<td>1935</td>
<td>Lawyer</td>
<td>Republican</td>
<td>Vermont</td>
<td>Harvard</td>
<td>English</td>
</tr>
<tr>
<td>William McKinley</td>
<td>1843</td>
<td>1901</td>
<td>Engineer</td>
<td>Republican</td>
<td>Iowa</td>
<td>Illinois</td>
<td>English</td>
</tr>
<tr>
<td>Sara 1874</td>
<td>1900</td>
<td>1891</td>
<td>Engineer</td>
<td>Republican</td>
<td>Iowa</td>
<td>Stanford</td>
<td>English</td>
</tr>
<tr>
<td>Elizabeth</td>
<td>1845</td>
<td>1919</td>
<td>Clothier</td>
<td>Republican</td>
<td>Missouri</td>
<td>Pennsylvania</td>
<td>Missouri</td>
</tr>
<tr>
<td>Samuel May</td>
<td>1849</td>
<td>1905</td>
<td>Senator</td>
<td>Republican</td>
<td>Missouri</td>
<td>Missouri</td>
<td>Missouri</td>
</tr>
<tr>
<td>John 1879</td>
<td>1896</td>
<td>1910</td>
<td>Soldier</td>
<td>Republican</td>
<td>Texas</td>
<td>Texas</td>
<td>Texas</td>
</tr>
<tr>
<td>Sarah 1877</td>
<td>1890</td>
<td>1930</td>
<td>Senator</td>
<td>Republican</td>
<td>New York</td>
<td>New York</td>
<td>New York</td>
</tr>
<tr>
<td>Jonathan</td>
<td>1850</td>
<td>1929</td>
<td>Senator</td>
<td>Republican</td>
<td>Missouri</td>
<td>Missouri</td>
<td>Missouri</td>
</tr>
<tr>
<td>John 1895</td>
<td>1920</td>
<td>1910</td>
<td>Senator</td>
<td>Republican</td>
<td>Missouri</td>
<td>Missouri</td>
<td>Missouri</td>
</tr>
<tr>
<td>Ann 1870</td>
<td>1924</td>
<td>1910</td>
<td>Senator</td>
<td>Republican</td>
<td>Missouri</td>
<td>Missouri</td>
<td>Missouri</td>
</tr>
<tr>
<td>State</td>
<td>Election Year</td>
<td>Party</td>
<td>Candidate Name</td>
<td>County</td>
<td>Admin</td>
<td>Location</td>
<td>Date</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>----------------</td>
<td>----------------</td>
<td>------------</td>
<td>--------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>Thurmond</td>
<td>1952</td>
<td>Republican</td>
<td>Eisenhower</td>
<td>Kentucky</td>
<td>A2</td>
<td>Tennessee</td>
<td>March 1789</td>
</tr>
<tr>
<td>Stevenson</td>
<td>1956</td>
<td>Republican</td>
<td>Eisenhower</td>
<td>Kentucky</td>
<td>A2</td>
<td>Tennessee</td>
<td>March 1789</td>
</tr>
<tr>
<td>Nixon</td>
<td>1960</td>
<td>Democratic</td>
<td>Kennedy</td>
<td>Kentucky</td>
<td>A2</td>
<td>Tennessee</td>
<td>March 1789</td>
</tr>
<tr>
<td>Goldwater</td>
<td>1964</td>
<td>Republican</td>
<td>Johnson</td>
<td>Kentucky</td>
<td>A2</td>
<td>Tennessee</td>
<td>March 1789</td>
</tr>
<tr>
<td>Adams</td>
<td>1793</td>
<td>Vermont</td>
<td>Adams</td>
<td>Vermont</td>
<td>A1</td>
<td>USA</td>
<td>March 1793</td>
</tr>
<tr>
<td>Ohio</td>
<td>1801</td>
<td>Jefferson</td>
<td>Adams</td>
<td>Vermont</td>
<td>A1</td>
<td>USA</td>
<td>March 1801</td>
</tr>
<tr>
<td>Louisiana</td>
<td>1809</td>
<td>Madison</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1809</td>
</tr>
<tr>
<td>Indiana</td>
<td>1813</td>
<td>Madison</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1813</td>
</tr>
<tr>
<td>Elbridge</td>
<td>1817</td>
<td>Indiana</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1817</td>
</tr>
<tr>
<td>Monroe</td>
<td>1821</td>
<td>Monroe</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1821</td>
</tr>
<tr>
<td>Alabama</td>
<td>1825</td>
<td>Michigan</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1825</td>
</tr>
<tr>
<td>Tennessee</td>
<td>1833</td>
<td>Michigan</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1833</td>
</tr>
<tr>
<td>Texas</td>
<td>1849</td>
<td>Michigan</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1849</td>
</tr>
<tr>
<td>Taylor</td>
<td>1850</td>
<td>Texas</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1850</td>
</tr>
<tr>
<td>Fillmore</td>
<td>1853</td>
<td>California</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1853</td>
</tr>
<tr>
<td>Pierce</td>
<td>1857</td>
<td>California</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1857</td>
</tr>
<tr>
<td>Buchanan</td>
<td>1865</td>
<td>Illinois</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1865</td>
</tr>
<tr>
<td>Kansas</td>
<td>1869</td>
<td>Oregon</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1869</td>
</tr>
<tr>
<td>Lincoln</td>
<td>1873</td>
<td>Kansas</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1873</td>
</tr>
<tr>
<td>Johnson</td>
<td>1875</td>
<td>Wisconsin</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1875</td>
</tr>
<tr>
<td>Grant</td>
<td>1878</td>
<td>Idaho</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1878</td>
</tr>
<tr>
<td>Cleavelan</td>
<td>1882</td>
<td>Washington</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1882</td>
</tr>
<tr>
<td>Harrisob</td>
<td>1885</td>
<td>Montana</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1885</td>
</tr>
<tr>
<td>Northd</td>
<td>1891</td>
<td>Wyoming</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1891</td>
</tr>
<tr>
<td>A33</td>
<td>March 1891</td>
<td>Idaho</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1893</td>
</tr>
<tr>
<td>A34</td>
<td>March 1893</td>
<td>Idaho</td>
<td>Adams</td>
<td>USA</td>
<td>A1</td>
<td>USA</td>
<td>March 1893</td>
</tr>
<tr>
<td>WHIG</td>
<td>25</td>
<td>2DEMOCRATIC</td>
<td>145WHIG</td>
<td>98CONGRESS</td>
<td>C25</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>30WHIG</td>
<td>108CONGRESS</td>
<td>2DEMETRIC</td>
<td>2WHIG</td>
<td>108WHIG</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>107OTHERS</td>
<td>2</td>
<td>3DEMOCRATIC</td>
<td>02WHIG</td>
<td>133OTHERS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>WHIG</td>
<td>280OTHERS</td>
<td>2</td>
<td>3DEMOCRATIC</td>
<td>25WHIG</td>
<td>28OTHERS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>6CONGRESS</td>
<td>79OTHERS</td>
<td>1CONGRESS</td>
<td>29</td>
<td>77OTHERS</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>142WHIG</td>
<td>25</td>
<td>3DEMOCRATIC</td>
<td>143WHIG</td>
<td>21OTHERS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>6CONGRESS</td>
<td>36WHIG</td>
<td>4CONGRESS</td>
<td>C31</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>30WHIG</td>
<td>9CONGRESS</td>
<td>38WHIG</td>
<td>5CONGRESS</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPUBLICAN</td>
<td>159WHIG</td>
<td>91OTHERS</td>
<td>4CONGRESS</td>
<td>38</td>
<td>3DEMOCRATIC</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>REPUBLICAN</td>
<td>143CONGRESS</td>
<td>3DEMOCRATIC</td>
<td>83REPUBLICAN</td>
<td>108OTHERS</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>10REPUBLICAN</td>
<td>118CONGRESS</td>
<td>36REPUBLICAN</td>
<td>200OTHERS</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OTHERS</td>
<td>9REPUBLICAN</td>
<td>114OTHERS</td>
<td>3DEMOCRATIC</td>
<td>92REPUBLICAN</td>
<td>114OTHERS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>10REPUBLICAN</td>
<td>9CONGRESS</td>
<td>3DEMOCRATIC</td>
<td>9REPUBLICAN</td>
<td>114OTHERS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>C41</td>
<td>2DEMETRIC</td>
<td>149CONGRESS</td>
<td>2DEMOCRATIC</td>
<td>3DEMOCRATIC</td>
<td>149CONGRESS</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>REPUBLICAN</td>
<td>42OTHERS</td>
<td>14CONGRESS</td>
<td>3DEMOCRATIC</td>
<td>2DEMOCRATIC</td>
<td>14CONGRESS</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>36REPUBLICAN</td>
<td>42</td>
<td>2DEMOCRATIC</td>
<td>2DEMOCRATIC</td>
<td>2DEMOCRATIC</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>14CONGRESS</td>
<td>149CONGRESS</td>
<td>3DEMOCRATIC</td>
<td>149CONGRESS</td>
<td>3DEMOCRATIC</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>OTHERS</td>
<td>37CONGRESS</td>
<td>141OTHERS</td>
<td>3DEMOCRATIC</td>
<td>141OTHERS</td>
<td>141OTHERS</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>REPUBLICAN</td>
<td>147OTHERS</td>
<td>11CONGRESS</td>
<td>3DEMOCRATIC</td>
<td>135REPUBLICAN</td>
<td>147OTHERS</td>
<td>147OTHERS</td>
<td>2</td>
</tr>
<tr>
<td>3DEMOCRATIC</td>
<td>169REPUBLICAN</td>
<td>152OTHERS</td>
<td>4CONGRESS</td>
<td>38</td>
<td>14CONGRESS</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DEMOCRATIC</td>
<td>44REPUBLICAN</td>
<td>3DEMOCRATIC</td>
<td>3DEMOCRATIC</td>
<td>3DEMOCRATIC</td>
<td>3DEMOCRATIC</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3DEMOCRATIC</td>
<td>159REPUBLICAN</td>
<td>159REPUBLICAN</td>
<td>159REPUBLICAN</td>
<td>159REPUBLICAN</td>
<td>159REPUBLICAN</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3DEMOCRATIC</td>
<td>3DEMOCRATIC</td>
<td>3DEMOCRATIC</td>
<td>3DEMOCRATIC</td>
<td>3DEMOCRATIC</td>
<td>3DEMOCRATIC</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>Republican</td>
<td>Democratic</td>
<td>Independent</td>
<td>Other</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>------------</td>
<td>-------------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arizona</td>
<td>1836</td>
<td>53104</td>
<td>28</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>California</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Colorado</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connecticut</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delaware</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Florida</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Georgia</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hawaii</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idaho</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Illinois</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indiana</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iowa</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kansas</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Louisiana</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maine</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maryland</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Massachusetts</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Michigan</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minnesota</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Missouri</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Montana</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nebraska</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nevada</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New York</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Jersey</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Mexico</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New York</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ohio</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oklahoma</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oregon</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pennsylvania</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rhode Island</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>South Dakota</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tennessee</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Texas</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Utah</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virginia</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Washington</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>West Virginia</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wisconsin</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wyoming</td>
<td>12892</td>
<td>12530</td>
<td>202000</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B

Presidential Report Examples
** PRESIDENT/ELECTION REPORT **

PRESIDENT NAME JOHN ADAMS
WON ELECTION OF 1796 AGAINST JEFFERSON

PRESIDENT NAME JOHN Q. ADAMS
WON ELECTION OF 1824 AGAINST JACKSON

PRESIDENT NAME CHESTER A. ARTHUR

PRESIDENT NAME JAMES BUCHANAN
WON ELECTION OF 1856 AGAINST FREMONT

PRESIDENT NAME GROVER CLEVELAND
WON ELECTION OF 1884 AGAINST BLAINE
WON ELECTION OF 1892 AGAINST HARRISON

PRESIDENT NAME CALVIN COOLIDGE
WON ELECTION OF 1924 AGAINST DAVIS

PRESIDENT NAME DWIGHT D. EISENHUWER
WON ELECTION OF 1952 AGAINST STEVENSON
WON ELECTION OF 1956 AGAINST STEVENSON

PRESIDENT NAME MILLARD FILLMORE

PRESIDENT NAME JAMES A. GARFIELD
WON ELECTION OF 1880 AGAINST HANCOCK

PRESIDENT NAME ULYSSES S. GRANT
WON ELECTION OF 1868 AGAINST SEYMOUR
**PRESIDENT/ELECTION REPORT**

President Name Warren
G. Harding
Won election of 1920 against Cox

President Name Benjamin
Harrison
Won election of 1888 against Cleveland

President Name William
H. Harrison
Won election of 1840 against Van Buren

President Name Rutherford
B. Hayes
Won election of 1876 against Tilden

President Name Herbert
C. Hoover
Won election of 1928 against Smith

President Name Andrew
Jackson
Won election of 1828 against Adams
Won election of 1832 against Clay

President Name Thomas
Jefferson
Won election of 1800 against Burr
Won election of 1804 against Pinckney

President Name Andrew
Johnson

President Name Lyndon
B. Johnson
Won election of 1964 against Goldwater

President Name John
F. Kennedy
Won election of 1960 against Nixon
<table>
<thead>
<tr>
<th>President Name</th>
<th>Election Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abraham Lincoln</td>
<td>Won election of 1860 against Douglas</td>
</tr>
<tr>
<td></td>
<td>Won election of 1864 against McClellan</td>
</tr>
<tr>
<td>James Madison</td>
<td>Won election of 1808 against Pinckney</td>
</tr>
<tr>
<td></td>
<td>Won election of 1812 against Clinton</td>
</tr>
<tr>
<td>William McKinley</td>
<td>Won election of 1896 against Bryan</td>
</tr>
<tr>
<td></td>
<td>Won election of 1900 against Bryan</td>
</tr>
<tr>
<td>James Monroe</td>
<td>Won election of 1816 against King</td>
</tr>
<tr>
<td></td>
<td>Won election of 1820 against Adams</td>
</tr>
<tr>
<td>Franklin Pierce</td>
<td>Won election of 1852 against Scott</td>
</tr>
<tr>
<td>James K. Polk</td>
<td>Won election of 1844 against Clay</td>
</tr>
<tr>
<td>Franklin D. Roosevelt</td>
<td>Won election of 1932 against Hoover</td>
</tr>
<tr>
<td></td>
<td>Won election of 1936 against Landon</td>
</tr>
<tr>
<td></td>
<td>Won election of 1940 against Willkie</td>
</tr>
<tr>
<td></td>
<td>Won election of 1944 against Dewey</td>
</tr>
<tr>
<td>Theodore Roosevelt</td>
<td>Won election of 1904 against Parker</td>
</tr>
</tbody>
</table>

**President/Election Report**

UNIVERSITY OF MICHIGAN / MTS

SEP 29, 1974 17:17.10 PAGE 3
PRESIDENT NAME WILLIAM H. TAFT
WON ELECTION OF 1908 AGAINST BRYAN

PRESIDENT NAME ZACHARY TAYLOR
WON ELECTION OF 1848 AGAINST CASS

PRESIDENT NAME HARRY S. TRUMAN
WON ELECTION OF 1948 AGAINST DEWEY

PRESIDENT NAME JOHN TYLER

PRESIDENT NAME MARTIN VAN BUREN
WON ELECTION OF 1836 AGAINST HARRISON

PRESIDENT NAME GEORGE WASHINGTON
WON ELECTION OF 1789 AGAINST ADAMS
WON ELECTION OF 1792 AGAINST ADAMS

PRESIDENT NAME WOODROW WILSON
WON ELECTION OF 1912 AGAINST ROOSEVELT
WON ELECTION OF 1916 AGAINST HUGHES
** STATE AND CITY REPORT **

**STATE NAME - ALABAMA**
- Admitted in the Year: 1819
- Major Cities Are:
  - Population: 340887
  - Mobile
  - Montgomery: 134393
  - Huntsville: 123519

**STATE NAME - ALASKA**
- Admitted in the Year: 1959

**STATE NAME - ARIZONA**
- Admitted in the Year: 1912
- Major Cities Are:
  - Population: 505666
  - Phoenix
  - Tucson: 236877

**STATE NAME - ARKANSAS**
- Admitted in the Year: 1836
- Major Cities Are:
  - Population: 128929
  - Little Rock

**STATE NAME - CALIFORNIA**
- Admitted in the Year: 1850
- Major Cities Are:
  - Population: 2479015
  - Los Angeles
  - San Francisco: 740316
  - San Diego: 573224
  - Oakland: 367548
  - Long Beach: 344168
  - Sacramento: 237712
  - San Jose: 204196
  - Fresno: 133929
  - Glendale: 119422
BERKELEY  111268
ANAHEIM  104184
TORRANCE  100991
SANTA ANA  100350

STATE NAME - COLORADO  ADMITTED IN THE YEAR 1876
MAJOR CITIES ARE POPULATION
DENVER  493887

STATE NAME - CONN.  ADMITTED IN THE YEAR 1788
MAJOR CITIES ARE POPULATION
HARTFORD  162178
BRIDGEPORT  156748
NEW HAVEN  141752
WATERBURY  107130

STATE NAME - DELAWARE  ADMITTED IN THE YEAR 1787

STATE NAME - FLORIDA  ADMITTED IN THE YEAR 1845
MAJOR CITIES ARE POPULATION
MIAMI  291688
TAMPA  274970
JACKSONV.  201030
ST. PETER.  181298

STATE NAME - GEORGIA  ADMITTED IN THE YEAR 1788
MAJOR CITIES ARE POPULATION
ATLANTA  487455
SAVANNAH  149245
STATE NAME: HAWAII
ADMITTED IN THE YEAR: 1959
MAJOR CITIES ARE: HONOLULU
POPULATION: 294194

STATE NAME: IDAHO
ADMITTED IN THE YEAR: 1890

STATE NAME: ILLINOIS
ADMITTED IN THE YEAR: 1818
MAJOR CITIES ARE: CHICAGO
POPULATION: 3550404
ROCKFORD
132109
PEORIA
103162

STATE NAME: INDIANA
ADMITTED IN THE YEAR: 1816
MAJOR CITIES ARE: INDIANAPOLIS
POPULATION: 476258
GARY
178320
FORTWAYNE
172594
EVANSVILLE
144463
SOUTH BEND
132445
HAMMOND
111698

STATE NAME: IOWA
ADMITTED IN THE YEAR: 1846
MAJOR CITIES ARE: DES MOINES
POPULATION: 206739
CEDAR RAPIDS
103545
** STATE AND CITY REPORT **

<table>
<thead>
<tr>
<th>State Name</th>
<th>Year Admitted</th>
<th>Major Cities Are</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kansas</td>
<td>1861</td>
<td>Wichita, Kansas City, Topeka</td>
<td>254698, 121901, 119484</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State Name</th>
<th>Year Admitted</th>
<th>Major Cities Are</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kentucky</td>
<td>1792</td>
<td>Louisville</td>
<td>390639</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State Name</th>
<th>Year Admitted</th>
<th>Major Cities Are</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Louisiana</td>
<td>1812</td>
<td>New Orleans, Shreveport, Baton Rouge</td>
<td>627525, 160535, 154190</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State Name</th>
<th>Year Admitted</th>
<th>Major Cities Are</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maine</td>
<td>1820</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State Name</th>
<th>Year Admitted</th>
<th>Major Cities Are</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maryland</td>
<td>1788</td>
<td>Baltimore</td>
<td>939024</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State Name</th>
<th>Year Admitted</th>
<th>Major Cities Are</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Massachusetts</td>
<td>1788</td>
<td>Boston, Worcester, Springfield, Cambridge</td>
<td>697197, 186587, 174463, 107716</td>
</tr>
<tr>
<td>State Name</td>
<td>Admitted in the Year</td>
<td>Major Cities Are</td>
<td>Population</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------</td>
<td>-------------------</td>
<td>------------</td>
</tr>
<tr>
<td>New Bedford</td>
<td>1837</td>
<td></td>
<td>102447</td>
</tr>
<tr>
<td>State Name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Michigan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minnesota</td>
<td>1858</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Missouri</td>
<td>1817</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Montana</td>
<td>1889</td>
<td></td>
<td></td>
</tr>
<tr>
<td>St. Paul</td>
<td>482872</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duluth</td>
<td>313411</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jackson</td>
<td>106884</td>
<td></td>
<td></td>
</tr>
<tr>
<td>St. Louis</td>
<td>750026</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kansas City</td>
<td>475539</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jackson</td>
<td>144422</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
** STATE AND CITY REPORT **

<table>
<thead>
<tr>
<th>State Name</th>
<th>Admitted In The Year</th>
<th>Major Cities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nevada</td>
<td>1864</td>
<td></td>
</tr>
<tr>
<td>New Hampshire</td>
<td>1788</td>
<td></td>
</tr>
<tr>
<td>New Jersey</td>
<td>1787</td>
<td>Newark 405220, Jersey City 276101, Paterson 143663, Camden 117159, Trenton 114167, Elizabeth 107698</td>
</tr>
<tr>
<td>New Mexico</td>
<td>1912</td>
<td></td>
</tr>
<tr>
<td>New York</td>
<td>1788</td>
<td>New York 7781984, Buffalo 532759, Rochester 318611, Syracuse 216038, Yonkers 190634</td>
</tr>
</tbody>
</table>
**STATE AND CITY REPORT**

**ALBANY** 129726
**NIAG. FALL** 102394
**UTICA** 100410

**STATE NAME: N.C.**
**ADMITTED IN THE YEAR** 1789
**MAJOR CITIES ARE** POPULATION
**CHARLOTTE** 201564
**GREENSBORO** 119574
**WIN. SALEM** 111135
**RALEIGH** 105722

**STATE NAME: N.D.**
**ADMITTED IN THE YEAR** 1889

**STATE NAME: OHIO**
**ADMITTED IN THE YEAR** 1803
**MAJOR CITIES ARE** POPULATION
**CLEVELAND** 810858
**CINCINNATI** 505550
**COLUMBUS** 471316
**TOLEDO** 318003
**AKRON** 290351
**DAYTON** 262332
**YOUNGSTOWN** 166689
**CANTON** 113631

**STATE NAME: OKLAHOMA**
**ADMITTED IN THE YEAR** 1907
**MAJOR CITIES ARE** POPULATION
**OKLA. CITY** 324253
**TULSA** 261685

**STATE NAME: OREGON**
<table>
<thead>
<tr>
<th>State Name</th>
<th>Admitted In The Year</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pennsylvania</td>
<td>1787</td>
<td>604332</td>
</tr>
<tr>
<td></td>
<td>Portland</td>
<td>372676</td>
</tr>
<tr>
<td>Rhode Island</td>
<td>1790</td>
<td>207498</td>
</tr>
<tr>
<td>South Carolina</td>
<td>1788</td>
<td></td>
</tr>
<tr>
<td>South Dakota</td>
<td>1889</td>
<td></td>
</tr>
<tr>
<td>Tennessee</td>
<td>1796</td>
<td>536585</td>
</tr>
<tr>
<td></td>
<td>Memphis</td>
<td>170874</td>
</tr>
<tr>
<td></td>
<td>Nashville</td>
<td>130009</td>
</tr>
<tr>
<td></td>
<td>Chattanooga</td>
<td>118827</td>
</tr>
<tr>
<td>Texas</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
** STATE AND CITY REPORT **

<table>
<thead>
<tr>
<th>State Name</th>
<th>Admitted In The Year</th>
<th>Major Cities Are</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utah</td>
<td>1896</td>
<td>S. L. City</td>
<td>189454</td>
</tr>
<tr>
<td>Vermont</td>
<td>1791</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virginia</td>
<td>1788</td>
<td>Norfolk</td>
<td>304869</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Richmond</td>
<td>219958</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Portsmouth</td>
<td>144773</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Newp. News</td>
<td>113662</td>
</tr>
<tr>
<td>Washington</td>
<td>1889</td>
<td>Seattle</td>
<td>557087</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TACOMA 147979

STATE NAME: W.VA
ADMITTED IN THE YEAR 1863

STATE NAME: WISCONSIN
ADMITTED IN THE YEAR 1848
MAJOR CITIES ARE
MILWAUKEE 741324
MADISON 157844

STATE NAME: WYOMING
ADMITTED IN THE YEAR 1890
PRESIDENT- ADAMS
WAS MARRIED IN 1764 TO ABIGAIL
HEADED ADMINISTRATIONS VICE PRESIDENT
IN 1797 JEFFERSON
BORN IN: MASS.

PRESIDENT- ADAMS
WAS MARRIED IN 1797 TO LOUISA
HEADED ADMINISTRATIONS VICE PRESIDENT
IN 1825 CALHOUN
BORN IN: MASS.

PRESIDENT- ARTHUR
WAS MARRIED IN 1859 TO ELLEN
HEADED ADMINISTRATIONS VICE PRESIDENT
IN 1881
BORN IN: VERMONT

PRESIDENT- BUCHANAN
WAS NOT MARRIED
HEADED ADMINISTRATIONS VICE PRESIDENT
IN 1857 BRECKRIDGE
BORN IN: PA.

PRESIDENT- CLEVELAND
WAS MARRIED IN 1886 TO FRANCES
HEADED ADMINISTRATIONS VICE PRESIDENT
IN 1885 HENDRICKS
IN 1893 STEVENSON
BORN IN: NEW JERSEY

PRESIDENT- COOLIDGE
WAS MARRIED IN 1905 TO GRACE
**PRESIDENTIAL INFORMATION REPORT **

IN 1923
IN 1925
DAWES
BORN IN: VERMONT

PRESIDENT- EISENHOWER
WAS MARRIED IN 1916 TO MAMIE
HEADED ADMINISTRATIONS  VICE PRESIDENT
IN 1953  NIXON
IN 1957  NIXON
BORN IN: TEXAS

PRESIDENT- FILLMORE
WAS MARRIED IN 1826 TO ABIGAIL
WAS MARRIED IN 1858 TO CAROLINE
HEADED ADMINISTRATIONS  VICE PRESIDENT
IN 1850
BORN IN: NEW YORK

PRESIDENT- GARFIELD
WAS MARRIED IN 1858 TO LUCRETIA
HEADED ADMINISTRATIONS  VICE PRESIDENT
IN 1881  ARTHUR
BORN IN: OHIO

PRESIDENT- GRANT
WAS MARRIED IN 1848 TO JULIA
HEADED ADMINISTRATIONS  VICE PRESIDENT
IN 1869  COLFAX
IN 1873  WILSON
BORN IN: OHIO

PRESIDENT- HARDING
WAS MARRIED IN 1891 TO FLORENCE
HEADED ADMINISTRATIONS  VICE PRESIDENT
**PRESIDENTIAL INFORMATION REPORT**

IN 1921           COOLIDGE
BORN IN: OHIO

PRESIDENT- HARRISON
WAS MARRIED IN 1853 TO CAROLINE
WAS MARRIED IN 1896 TO MARY
HEADED ADMINISTRATIONS    VICE PRESIDENT
IN 1889              MORTON
BORN IN: OHIO

PRESIDENT- HARRISON
WAS MARRIED IN 1795 TO ANNA
HEADED ADMINISTRATIONS    VICE PRESIDENT
IN 1841              TYLER
BORN IN: VIRGINIA

PRESIDENT- HAYES
WAS MARRIED IN 1852 TO LUCY
HEADED ADMINISTRATIONS    VICE PRESIDENT
IN 1877              WHEELER
BORN IN: OHIO

PRESIDENT- HOOVER
WAS MARRIED IN 1899 TO LOU
HEADED ADMINISTRATIONS    VICE PRESIDENT
IN 1929              CURTIS
BORN IN: IOWA

PRESIDENT- JACKSON
WAS MARRIED IN 1791 TO RACHEL
HEADED ADMINISTRATIONS    VICE PRESIDENT
IN 1829              CALHOUN
IN 1833              VAN BUREN
BORN IN: S. C.
PRESIDENT- JEFFERSON
WAS MARRIED IN 1772 TO MARTHA
HEADED ADMINISTRATIONS  VICE PRESIDENT
   IN 1801  BURR
   IN 1805  CLINTON
BORN IN: VIRGINIA

PRESIDENT- JOHNSON
WAS MARRIED IN 1827 TO ELIZA
HEADED ADMINISTRATIONS  VICE PRESIDENT
   IN 1865
BORN IN: N.C.

PRESIDENT- JOHNSON
WAS MARRIED IN 1934 TO CLAUDIA
HEADED ADMINISTRATIONS  VICE PRESIDENT
   IN 1963
   IN 1965  HUMPHREY
BORN IN: TEXAS

PRESIDENT- KENNEDY
WAS MARRIED IN 1953 TO JACQUELINE
HEADED ADMINISTRATIONS  VICE PRESIDENT
   IN 1961  JOHNSON
BORN IN: MASS.

PRESIDENT- LINCOLN
WAS MARRIED IN 1842 TO MARY
HEADED ADMINISTRATIONS  VICE PRESIDENT
   IN 1861  HAMLIN
   IN 1865  JOHNSON
BORN IN: KENTUCKY

PRESIDENT- MADISON
WAS MARRIED IN 1794 TO DOLLEY
HEADED ADMINISTRATIONS	VICE PRESIDENT
IN 1809	CLINTON
IN 1813	GERRY
BORN IN: VIRGINIA

PRESIDENT—MCKINLEY
WAS MARRIED IN 1871 TO IDA
HEADED ADMINISTRATIONS	VICE PRESIDENT
IN 1897	HOBART
IN 1901	ROOSEVELT
BORN IN: OHIO

PRESIDENT—MONROE
WAS MARRIED IN 1786 TO ELIZABETH
HEADED ADMINISTRATIONS	VICE PRESIDENT
IN 1817	TOMPKINS
IN 1821	TOMPKINS
BORN IN: VIRGINIA

PRESIDENT—PIERCE
WAS MARRIED IN 1834 TO JANE
HEADED ADMINISTRATIONS	VICE PRESIDENT
IN 1853	KING
BORN IN: N.H.

PRESIDENT—POLK
WAS MARRIED IN 1824 TO SARAH
HEADED ADMINISTRATIONS	VICE PRESIDENT
IN 1845	DALLAS
BORN IN: N.C.

PRESIDENT—ROOSEVELT
WAS MARRIED IN 1805 TO ANN
HEADED ADMINISTRATIONS  VICE PRESIDENT
                 IN 1933       GARNER
                 IN 1937       GARNER
                 IN 1941       WALLACE
                 IN 1945       TRUMAN

BORN IN: NEW YORK

PRESIDENT- ROOSEVELT
WAS MARRIED IN 1880 TO ALICE
WAS MARRIED IN 1886 TO EDITH
HEADED ADMINISTRATIONS  VICE PRESIDENT
                 IN 1901       FAIRBANKS
                 IN 1905

BORN IN: NEW YORK

PRESIDENT- TAFT
WAS MARRIED IN 1886 TO HELEN
HEADED ADMINISTRATIONS  VICE PRESIDENT
                 IN 1909       SHERMAN

BORN IN: OHIO

PRESIDENT- TAYLOR
WAS MARRIED IN 1810 TO MARGARET
HEADED ADMINISTRATIONS  VICE PRESIDENT
                 IN 1849       FILLMORE

BORN IN: VIRGINIA

PRESIDENT- TRUMAN
WAS MARRIED IN 1919 TO ELIZABETH
HEADED ADMINISTRATIONS  VICE PRESIDENT
                 IN 1945
                 IN 1949       BARKLEY

BORN IN: MISSOURI
PRESIDENT - TYLER
WAS MARRIED IN 1813 TO LETITIA
WAS MARRIED IN 1844 TO JULIA
HEADED ADMINISTRATIONS VICE PRESIDENT
   IN 1841
BORN IN: VIRGINIA

PRESIDENT - VAN BUREN
WAS MARRIED IN 1807 TO HANNAH
HEADED ADMINISTRATIONS VICE PRESIDENT
   IN 1837  JOHNSON
BORN IN: NEW YORK

PRESIDENT - WASHINGTON
WAS MARRIED IN 1759 TO MARTHA
HEADED ADMINISTRATIONS VICE PRESIDENT
   IN 1789  ADAMS
   IN 1793  ADAMS
BORN IN: VIRGINIA

PRESIDENT - WILSON
WAS MARRIED IN 1885 TO ELLEN
WAS MARRIED IN 1915 TO EDITH
HEADED ADMINISTRATIONS VICE PRESIDENT
   IN 1913  MARSHALL
   IN 1917  MARSHALL
BORN IN: VIRGINIA