ABSTRACT

This project develops an expert system to support the private company, PSYCHOLOGICAL RESOURCES OF DALLAS. The system accepts input of personal data and answers from a questionnaire, and stores the data in a file of records that may be used to provide: an evaluation of test results, a finished letter report, and a work sheet documenting germane information together with case-specific actions to be taken by clerical personnel. The system also provides methods for norming and comparing answers supplied to the questionnaire.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>A-1</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>C-1</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>I-1</td>
</tr>
<tr>
<td>NARRATIVE</td>
<td>N-1</td>
</tr>
<tr>
<td>Through N-7</td>
<td></td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>P-1</td>
</tr>
<tr>
<td>Through P-50</td>
<td></td>
</tr>
<tr>
<td>RESULTS AND CONCLUSIONS</td>
<td>R-1</td>
</tr>
<tr>
<td>Through R-3</td>
<td></td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>BB-1</td>
</tr>
<tr>
<td>APPENDICES</td>
<td></td>
</tr>
<tr>
<td>APPENDIX A Listings</td>
<td></td>
</tr>
<tr>
<td>PLUGFORM</td>
<td>PF-1</td>
</tr>
<tr>
<td>Through PF-19</td>
<td></td>
</tr>
<tr>
<td>UNITGLOBALS</td>
<td>GLOBALS-1</td>
</tr>
<tr>
<td>Through GLOBALS-3</td>
<td></td>
</tr>
<tr>
<td>UNIT READER</td>
<td>READER-1</td>
</tr>
<tr>
<td>Through READER-15</td>
<td></td>
</tr>
<tr>
<td>UNIT GETSPUTS</td>
<td>GETSPUTS-1</td>
</tr>
<tr>
<td>Through GETSPUTS-29</td>
<td></td>
</tr>
<tr>
<td>UNIT DATMANIP</td>
<td>DATMANIP-1</td>
</tr>
<tr>
<td>Through DATMANIP-20</td>
<td></td>
</tr>
<tr>
<td>WordPerfect Macros</td>
<td>WPM-1</td>
</tr>
<tr>
<td>Through WPM-19</td>
<td></td>
</tr>
<tr>
<td>APPENDIX B Results</td>
<td>B-1</td>
</tr>
<tr>
<td>Through B-36</td>
<td></td>
</tr>
<tr>
<td>APPENDIX C Letter from</td>
<td></td>
</tr>
<tr>
<td>PSYCHOLOGICAL RESOURCES OF DALLAS</td>
<td></td>
</tr>
</tbody>
</table>
INTRODUCTION

PSYCHOLOGICAL RESOURCES OF DALLAS was undertaking the development of written tests to identify and gauge the extent of certain psychological disorders. To support the development and administration of these tests, as well as their update and maintenance, the company desired an integrated, tailored expert system. As the test on which this project was based was original, there were no commercially available software packages which could fill the company's requirements. Additional limitations which mediated for a tailored system consisted of the company's disinclination to update or replace their hardware, disk operating systems, and word processing software (i.e., 80286 personal computers, MS-DOS 4.01 and 5.0, and WordPerfect 5.1.)

In a tour of duty at the Pentagon, 1987-1991, I developed systems for accepting minimal (mostly "yes or no") data and composing highly specific letters responding to requests. My work resulted in a reduction of average letter composition times by approximately 95 percent. Programs which evaluate test results, write letters and instructions, or normalize data are legion. If there were systems which would do all three, however, they would only support specific tests. The specificity of the test developed, the nature of the letters required, and other requirements of the company ensured the uniqueness of the project.

The project covered the spectrum from personal experience through the bulk of the individual courses taken in the master's degree program in Computer Science, including courses in business information systems, artificial intelligence, expert systems, programming, database management, et al. The problem was by its nature unique. It was rigorous and it integrated the major skills required for the desired degree.

The software system required to complete this project was a management information system with an expert system component. Its objectives were to satisfy the needs of upper-management personnel and remove them from those decisions which were foregone conclusions, to ensure the participation of top-level management in those decisions where their judgement was required, to increase the productivity of personnel performing rote and repetitive tasks, to protect against errors, and to enhance the overall efficiency of the entire company team. These objectives were identical to those of numerous projects in which I was involved during 30 years of military service.
NARRATIVE

The purpose of this project was to provide a software system, including an imbedded expert system, to support the private company, PSYCHOLOGICAL RESOURCES OF DALLAS, hereafter referred to as "the company", in the evaluation and administration of psychological test results. The psychological test (survey) involved was the Post Accident Stress Survey (PASS). The software system developed is called PLUGFORM.

Upon execution of the software system, the Main Menu is displayed. From this menu the user is able to:

1) Access an explanation of the various systems options.
2) Enter data to compose new records, letter files, and work sheets.
3) Access existing records to reconstruct response letters and work sheets.
5) Access a menu for norming and comparing record files and individual records.
6) Exit the PLUGFORM system.

Option 1 accesses a 15 screen, step by step explanation of what the user may expect to encounter while executing any of the program options. The users may also target germane sections of this explanation directly from the menus accessed through options 4 and 5.

Option 2 is specifically designed for clerical personnel. By inputting data, in response to unequivocal prompts, the user creates in-house work sheets, client response letters, and a record entry to the Working Records File, which is used in all other working options.

Option 2 input to the software system comes from two standard forms, one of which is a referral (from either an attorney or a physician) and the other is a separately administered psychological survey. This data is entered via the keyboard. Captured information consists of:

1) The field (medicine or law) of the referring client.
2) The name of the person referred for testing.
3) The date on which the referred person's test is scored.
4) The referred person's answers to the 35 test questions.
5) The address of the referring client.
6) The sex of the person referred for testing.
7) The date of birth of the person referred for testing.
8) The date of the accident precipitating the test.
9) The date on which the test results are received.
10) The stated condition or indication of head trauma.
11) The stated condition of emotional trauma.
12) A description of the precipitating accident, if available.
13) The salutation used in the response letter.
14) Additional symptoms mentioned on the referral form.
15) An automatically added record number.

The system captures the above information in the same record format in which it is stored. After the survey answers are input, they are displayed to the screen and the user is provided the opportunity to correct them. Obvious errors (e.g., nonnumeric entries or values outside the allowable numerical range) result in a screen notice to the user. After the answers and personal data have been input, the system creates the in-house worksheet, a file holding the response letter, and a screen display of the evaluation results. A record containing the input is then appended to the Working Records File and the tested person's name is appended to the Names File.

The imbedded expert system manipulates weighted values of various combinations of answers and compares the results to threshold moduli. These comparisons determine if particular disorders or characteristics are indicated at high, moderate, low, or negligible levels. It then combines the various levels and reaches a determination regarding indications of psychological disturbance. These calculated values are incorporated into both the work sheet and the response letter. The expert system also chooses a final summation paragraph using a matrix of values, some entered directly and some calculated from other input data.

Examples of generated work sheets and response letters are found in the APPENDICES.

The user is provided an option of bypassing the printing of the work sheet. After that choice is made, control returns to the Main Menu. Each response letter is written to the floppy disk as a separate file by the last name of the person tested (or the first eight letters of the last name), with a suffix of his first two initials, and an "L" (for letter file). Letters are completed from WordPerfect by use of WordPerfect macros, which are included in the software system. A first single letter macro calls up all floppy disk response letter files. After retrieving the proper file, the user enters a second single letter macro which is chained to a series of other macros. These macros complete the response letter and leave the user in the WordPerfect Speller. Somewhat more detailed information regarding the work sheet and response letter are contained at the addendum.
On entering Option 3, the user is asked to input the complete name of the person whose record is desired. That name is used as a search key and, upon matching, the completed answer template is displayed to the screen and, excepting appending records, the same post input sequence of events outlined for Option 2 occurs. This obviates the need to maintain hard copy records or documents, or may serve as a backup to more traditional filing systems.

Option 4 shifts control to the Record Manipulation Menu, containing options to:

1) Correct recorded survey answers.
2) Delete user specified records.
3) Sort records on user specified fields.
4) Provide an alphabetical listing of relevant data.
5) Access the Record Manipulation explanation.
6) Return to the Main Menu.

Record Manipulation Menu Option 1 is similar to Main Menu Option 3. They differ only in the fact that the Record Manipulation option deletes the old record and appends the corrected record to the Working Records File.

Record Manipulation Menu Option 2 deletes the specified record from both the Working Records File and the Names File.

Record Manipulation Menu Option 3 provides a screen listing of the Name, Sex, Date of Birth, Date of Accident, Date of Receipt, Head Trauma, and Psychological Trauma, sorted by the user's choice of dates.

Record Manipulation Menu Option 4 provides the same information as Option 3, sorted alphabetically. The user may choose a screen display or a printed copy.

Main Menu Option 5 shifts control to the Norming and Comparing Menu, which contains options to provide for:

1) The creation of user named norming files.
2) The normalization of files.
3) The display of normalized files.
4) Display of normed information against a specified record.
5) Display of normed information against the current record.
6) Input to established norming files.
7) Access to the Norming and Comparing explanation.
8) Return to the Main Menu.
Norming and Comparing Option 1 prompts the user to name a norming file. It then asks, successively, if personnel names, today's date, dates of birth, and dates of testing are required. If so, they are prompted for and captured during file building. If not, only the subject's sex and survey answers are prompted for, as the records are of personnel who have not been exposed to situations deemed traumatic. While creating files from surveys provided by the company, marked statistical differences were noted between the sexes, thus the prompt for the subject's sex is always to be used. Entries are made in the same manner as in Main Menu Option 2.

Norming and Comparing Option 2 prompts the user to name the file he wants to normalize. There are presently four files which the user may call by number: a male norming file with 88 legitimate entries, a female norming file with 132 legitimate entries, a combined file with those same 220 entries, and the Working Records File with more than 150 fictional entries. The user may also enter the name of any other existing file, such as one which may be made using the previous option.

Norming and Comparing Option 3 prompts the user to choose one of the four files mentioned above, or to name any existing file which has been normalized. The choice is displayed to the screen, listing the mean, the variance, and the standard deviation of weighted scores for Depression, Anxiety, Post Traumatic Stress Syndrome, as well as the number and percentage of positive answers in each category. Symptom exaggeration statistics are also provided. (Those questions seeking indications of neurological disorders, part of the original proposal, were removed from the survey by the company.) Data in this option is displayed to the tenth decimal place.

Norming and Comparing Option 4 prompts the user for the name of the person whose record is to be compared, then prompts for the normalized file to use for the comparison. This option compares the named person's survey scores in the evaluated categories against the mean, variance and standard deviation of the normalized file. It also calculates and displays the individual's Z scores against that particular file.

Norming and Comparing Option 5 does the same thing as Option 4, except that there is no prompt for an individual name. The comparison is made for the record most recently made or called from file during the session.
Norming and Comparing Option 6 prompts the user for the norming file he wants to augment. The program checks which optional data was captured for the named file and sets the building prompts accordingly. The category threshold moduli (between indications being negligible, low, moderate, or high) are calculated as a function of standard deviation added to the mean. They are based on the male and female norming files. In accordance with the company’s desire, they are static within the system. Additions made with this option do not change them, but they do change the results of Options 4 and 5 above, and can provide an early warning of a need to make adjustments. Norming and Comparing Options 1 and 6 are for clerical personnel. For those options falling between them, the users are Doctors of Psychology.
NARRATIVE  Addendum

The work sheet displays, directly from the data submitted:

1) The tested person's name.
2) The tested person's date of birth.
3) The date of the precipitating accident.
4) The date the test is received.
5) The attorney or physician involved.
6) An accident description (if given).
7) Any additional symptoms reported.
8) Whether or not head trauma is reported.
9) Whether or not emotional trauma is reported.
10) The answers to the 35 test questions.

The imbedded expert system also provides for a work sheet display of:

1) Whether the test results indicate a potential problem.
2) A table of interpretive scores on the disorders for which the test surveys.
3) An administrative table indicating:
   a) Who is to review the work sheet.
   b) What actions the clerical personnel are to take.

The choice between eight different summation paragraphs is based on a matrix made from the following criteria:

1. Psychological Trauma Evident Based on PASS
   Head Trauma Not Reported

2. Psychological Trauma Not Evident Based on PASS
   Psychological Trauma Evident Based on Accident Description
   Head Trauma Not Reported

3. Psychological Trauma Evident Based on PASS with Symptom Exaggeration
   Psychological Trauma Not Evident Based on Accident Description
   Head Trauma Not Reported

4. Psychological Trauma Not Evident Based on PASS
   Psychological Trauma Not Evident Based on Accident Description
   Head Trauma Not Reported

5. Psychological Trauma Evident Based on PASS
   Head Trauma Reported
6. Psychological Trauma Not Evident Based on PASS
   Psychological Trauma Evident Based on Accident Description
   Head Trauma Is Reported

7. Psychological Trauma Evident Based on PASS with Symptom
   Exaggeration
   Psychological Trauma Not Evident Based on Accident
   Description
   Head Trauma Is Reported

8. Psychological Trauma Not Evident Based on PASS
   Psychological Trauma Not Evident Based on Accident
   Description
   Head Trauma Is Reported

The last entry in any Response Letter File is either "AAAA", "BBBB", ..., or "HHHH", corresponding to the above. One macro, ALTV, is chained to a series of macros which format the letter, alter its phraseology to conform to the profession of the client, search until one of the eight "macro triggers" listed above is found and insert the corresponding paragraph into the letter, add the standard final paragraph and signature block, and send the letter into WordPerfect Spelling Check.
PROCEDURE

The system, called PLUGFORM, is built around two record types. The WORKING RECORD, composed of fields taken from keyboard input, is permanently stored. Fields of the CALCULATED VALUES RECORD contain the results of the program acting on the WORKING RECORD fields. The CALCULATED VALUES RECORD is used primarily as a vehicle for passing parameters between procedures, but is also used for writing to and reading from normalized files.

The WORKING RECORD is as follows:

<table>
<thead>
<tr>
<th>PRectype = record</th>
<th></th>
<th>WORKING RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rname : string[50]</td>
<td>(*)</td>
<td>Name of Person, Last First Middle (*)</td>
</tr>
<tr>
<td>Dori : char</td>
<td>(*)</td>
<td>For Doctor or Lawyer Client (*)</td>
</tr>
<tr>
<td>Pname : string[50]</td>
<td>(*)</td>
<td>Name of Person, First Middle Last (*)</td>
</tr>
<tr>
<td>Today : string[18]</td>
<td>(*)</td>
<td>Date for Letter/Test Scored on (*)</td>
</tr>
<tr>
<td>Qans : string[36]</td>
<td>(*)</td>
<td>Test Answers to the 35 questions (*)</td>
</tr>
<tr>
<td>CAddr : string</td>
<td>(*)</td>
<td>Address of Company’s Client (*)</td>
</tr>
<tr>
<td>Gender : string[6]</td>
<td>(*)</td>
<td>Sex of the Person Who Was Tested (*)</td>
</tr>
<tr>
<td>Bdate : string[8]</td>
<td>(*)</td>
<td>Date of Birth of Person Tested (*)</td>
</tr>
<tr>
<td>Adate : string[8]</td>
<td>(*)</td>
<td>Date of Precipitating Accident (*)</td>
</tr>
<tr>
<td>Rdate : string[8]</td>
<td>(*)</td>
<td>Date Request Received in Office (*)</td>
</tr>
<tr>
<td>HT : string[3]</td>
<td>(*)</td>
<td>Was Head Trauma Indicated? (*)</td>
</tr>
<tr>
<td>ADisc : string</td>
<td>(*)</td>
<td>Accident Description if Available (*)</td>
</tr>
<tr>
<td>Nsal : string[50]</td>
<td>(*)</td>
<td>Name for Salutation of Letter (*)</td>
</tr>
<tr>
<td>AdSymp : string</td>
<td>(*)</td>
<td>Additional Symptoms if Available (*)</td>
</tr>
<tr>
<td>Rnum : integer</td>
<td>(*)</td>
<td>Record Number - chronological (*)</td>
</tr>
</tbody>
</table>

This record requires 1022 bytes of memory.
The CALCULATED VALUES RECORD is as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>calctype</td>
<td>record</td>
<td></td>
</tr>
<tr>
<td>dstring</td>
<td>string[10]</td>
<td>(Verbal Score for Depression Disorder)</td>
</tr>
<tr>
<td>astring</td>
<td>string[10]</td>
<td>(Verbal Score for Anxiety Disorder)</td>
</tr>
<tr>
<td>pstring</td>
<td>string[10]</td>
<td>(Verbal Score for P.T. Stress Disorder)</td>
</tr>
<tr>
<td>estring</td>
<td>string[10]</td>
<td>(Verbal Score for Exaggeration)</td>
</tr>
<tr>
<td>teller</td>
<td>integer</td>
<td>(Sums indications for overall rating)</td>
</tr>
<tr>
<td>EX</td>
<td>integer</td>
<td>(Raw # of &quot;4&quot; answers, to catch faking)</td>
</tr>
<tr>
<td>D</td>
<td>real</td>
<td>(Weighted Depression Score)</td>
</tr>
<tr>
<td>DP</td>
<td>real</td>
<td>(Percent Depression Answers Positive)</td>
</tr>
<tr>
<td>D1</td>
<td>real</td>
<td>(Number of Depression Answers Positive)</td>
</tr>
<tr>
<td>A</td>
<td>real</td>
<td>(Weighted Anxiety Score)</td>
</tr>
<tr>
<td>AP</td>
<td>real</td>
<td>(Percent Anxiety Answers Positive)</td>
</tr>
<tr>
<td>A1</td>
<td>real</td>
<td>(Number of Anxiety Answers Positive)</td>
</tr>
<tr>
<td>P</td>
<td>real</td>
<td>(Weighted PTSD Score)</td>
</tr>
<tr>
<td>PP</td>
<td>real</td>
<td>(Percent PTSD Answers Positive)</td>
</tr>
<tr>
<td>P1</td>
<td>real</td>
<td>(Number of PTSD Answers Positive)</td>
</tr>
<tr>
<td>EXP</td>
<td>real</td>
<td>(Percent Exaggerated Answers)</td>
</tr>
<tr>
<td>EX1</td>
<td>real</td>
<td>(This Used Only for % Calculation)</td>
</tr>
</tbody>
</table>

This record requires 114 bytes of memory.

PLUGFORM consists of a main program, also named PLUGFORM, supported by four nonstandard units:

1) UNIT READER, which displays information regarding PLUGFORM and explanations on its operation.

2) UNIT GLOBALS, which contains the global variables and an initialization procedure.

3) UNIT GETSPUTS, which contains the bulk of the procedures used to obtain and write to file the data for the response letter.

4) UNIT DATMANIP, which contains the bulk of the computational procedures.

All external files, created or used by the system, are text files. They include:

1) The Working Records File. This file contains only keyboard input and can be used for reconstructing Work Sheets and Response Letter Files, or for running statistical comparisons on individuals.

2) The Names File. This file consists solely of the names, in last-first-middle order, of evaluated individuals and is updated at the same time as the Working Records File. It is used as an index file to shorten the times required to manipulate records.
3) Norming Files. These files are in the same format as the Working Records File and are written to and read from by use of Working Records. Three Norming Files (Male, Female, and Combined) contain survey responses from valid norming samples. They and the Working Records File may be called by number from several statistical procedures within the system. There is also a user option to name and build additional Norming Files.

4) Normed Data Files. Normalizing the data from any Norming File, creates a Normed Data File of the same name, but with a suffix of "nom". These files may be directly accessed from several statistical procedures within the system. These files are written to and read from via Calculated Values Records.

5) Response Letter Files. Each file is named for the individual being evaluated. Each contains the various component parts of an individual response letter; some standard, some containing unmodified keyboard input, and some modified or chosen wholly by the imbedded expert system. Additionally, each Response Letter File contains "triggers" for WordPerfect macros. These files are designed to produce Response Letters in WordPerfect. Once a response letter has been printed, its corresponding file may (indeed should) be deleted. It, as well as any related paper file, is redundant to the Working Records File. Test results, work sheets, and response letters may be reproduced in seconds from the Working Records File. The system has no interface with Response Letter Files, once they are created, but contains all the information required to reconstruct them.

There are also a series of 18 macros which may act upon the Response Letter Files from WordPerfect. One lists the Response Letter Files to the screen. Thirteen others are chained together and search the chosen file for "triggers" which determine the manner in which the individual response letter is completed. Four macros cover special situations. These macros are covered in more detail in the MODULE DEFINITIONS section below.

FLOW DIAGRAM THROUGH THE SYSTEM

On execution, the system will performs some "housekeeping" functions, initializing variables, loading an array from the Names File, sorting the array and rewriting the Names File. Those procedures (INITIALIZE, LOAD_NAME ARRAY, SHELPSORT, and REWRITE_NAMEFILE) are all shown in the MODULES DEFINITIONS section below. The first procedure visible to the user is PROCEDURE MAIN_MENU. Upon reaching MAIN_MENU, the system becomes interactive. Every menu option, or series of procedures, within the program, eventually returns to the menu from which it was called. Thus, one may only exit PLUGFORM via the Main Menu. In any one session, one may draft any number of response letters, reconstruct letters from working records, do statistics, etc., going from one menu to another at will.
MODULE DEFINITIONS

The program starts by running the four housekeeping functions shown below, which will be covered in more detail later, then calls MAIN_MENU, the first interface visible to the user.

BEGIN (* MAIN *)
    INITIALIZE (Cal);
    LOAD_NAME_ARRAY (Keep);
    SHELLSORT (Keep);
    REWRITE_NAMEFILE (Keep);
    MAIN_MENU (PR, Keep, Cal);
END.

A majority of the procedures which follow are simple and/or obvious and will only briefly be paraphrased. Those procedures which are more complex or which are more instructive regarding the overall design and implementation of the system, will be covered in more detail, up to the inclusion of the bulk of the source code.

In each comment section of the below listed procedures, following the RECEIVES/RETURNS remarks, a "genealogy" of the procedure is given. Every path between the MAIN MENU and the procedure is shown, with paths initiated by different options at the MAIN MENU, the RECORD MANIPULATION MENU, or the NORMING AND COMPARING MENU separated by slash marks. A dash after the MENU is followed by the option, with a period separating the procedure within that option; e.g., MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-8 would indicate that the procedure was the eighth procedure called from Procedure BUILD_NORM_FILE which, in turn, was the fourth procedure called from the first option of NORM_MENU, and that NORM_MENU was the fifth option of MAIN_MENU. When a procedure may be called multiple times from its preceding procedure, the calls are connected with ampersands; e.g., ... GetAnswers-2&4&7 would indicate that procedure is the second, fourth, and seventh procedure called from GetAnswers. When Procedure "C" may be called from both Procedure "A" and Procedure "B", and Procedure "B" may also be called from Procedure "A", the paths are separated by an "|"; e.g.; ... GetAnswers-4 | GetAnswers-6&9, GetAnsFast-2 would indicate that procedure was the fourth procedure called from GetAnswers and the second called from GetAnsFast, and that GetAnsFast was the sixth and ninth procedure called from GetAnswers. This notation system lists any preceding procedure which might call the annotated procedure, as a path back to the MAIN MENU; i.e., a conditional is treated as being satisfied. The last thing listed in the "comments" is a listing of all those procedures directly called from the annotated procedure.
PROCEDURE MAIN_MENU (VAR MR : PRectype; VAR TotNames : integer;
    VAR Cal : Calctype);
(* This procedure provides access to program operating instructions as *)
(* well as to the four major areas of program execution and the exit. *)

(* RECEIVES - a VAR Working Record, a VAR Calculated Values Record, and *)
(* an integer containing the total number of records in the *)
(* Working Records File. *)
(* RETURNS - access to any part of the system *)

(* MAIN-5 *)
(* Calls - ReadInst, NEW_LETTER, REC_LETTER, RECS_MENU, NORM_MENU *)

VAR
   MajorSet : char;  (* Input to the menu for access to program options *)

BEGIN
  REPEAT
    clrscr;
    WRITELN;
    WRITELN ('        WELCOME TO PLUGFORM   ');
    WRITELN ('        MAIN MENU    ');
    WRITELN (' Enter If You Wish To - ');
    WRITELN ('  1 Read about PLUGFORM. ');
    WRITELN ('  2 Compose a(nother) letter for a new case. ');
    WRITELN ('  3 Reconstruct a letter from an existing record. ');
    WRITELN ('  4 Visit RECORD MANIPULATION MENU. ');
    WRITELN ('  5 Visit NORMING AND COMPARING MENU. ');
    WRITELN ('  6 Quit PLUGFORM program. ');
    WRITELN;
    WRITE ('');
    READLN (MajorSet);
    clrscr;
    IF MajorSet = '1' THEN ReadInst (1);
    IF MajorSet = '2' THEN NEW_LETTER (MR, TotNames, Cal);
    IF MajorSet = '3' THEN REC_LETTER (MR, Cal);
    IF MajorSet = '4' THEN RECS_MENU (MR, TotNames);
    IF MajorSet = '5' THEN NORM_MENU (MR, Cal);
    UNTIL MajorSet = '6';
END; (* MAIN_MENU *)

P-6
PROCEDURE ReadInst (x : integer);
(* This procedure is intended to explain the operation of the program to *)
(* new users. Page control is circular and allows direct entry from *)
(* several locations within the main program. *)

(* RECEIVES – an integer variable representing a "page" of information *)
(* RETURNS – screens of information regarding PLUGFORM *)

(* MAIN_MENU-1  / *)
(* MAIN_MENU-4, RECS_MENU-5.1 / *)
(* MAIN_MENU-5, NORM_MENU-7.1 Calls - N/A *)

PROCEDURE NEW_LETTER (VAR NR : Prectype; VAR keep : integer; 
VAR Cal : CalcType);

(* This procedure calls the procedures required to score a test for the *)
(* first time, to draft an original letter to the client and write it to *)
(* a unique Response Letter file, to print a WorkSheet, and to insert a *)
(* record to the file WORKRECS.REC containing all captured information. *)

(* RECEIVES – a VAR Working Record, a VAR Calculated Values Record, and *)
(* a VAR integer containing the total number of records in *)
(* the Working Records File. *)
(* RETURNS – a completed Response Letter File, *)
(* a Working Record with all fields set, *)
(* a Calculated Values Record with all fields set, *)
(* a Work Sheet, and *)
(* an updated Working Records File. *)

(* MAIN MENU-2 *)
(* Calls – INITIALIZE, GetDorL, GetNames, GetToday, GetInAdd, GetGendr, *)
(* GetDates, GetTrama, GetAccid, GetSalut, GetAdSym, GetRspsFile, *)
(* PutHead, PutDate, PutInAdd, PutREfer, PutSalut, PutParal, GetAnswers, *)
(* GetMaleLimits or GetFemeLimits, PutEndLet, MAKE_Qans_STRING, MAKEREC, *)
(* MAKENAME, and WorkSheet *)

VAR

InName : string; (* Returns & passes name of Response Letter file*)
Continuing : boolean; (* Allows return to Main Menu for name duplicate*)
AnsGate : integer; (* A flag to calculating procedure GetAnswers *)
BEGIN
clrscr;
INITIALIZE (Cal);
Ansgate := 2;
NR.Rnum := keep + 1;
GetDorL (NR);
GetNames (NR, keep, Continuing, InName);
IF Continuing THEN
BEGIN
    GetToday (NR);
    GetInAddr (NR);
    GetGendr (NR);
    GetDates (NR);
    GetTrama (NR);
    GetAccid (NR);
    GetSalut (NR);
    GetAdSym (NR);
    GetRspFile (GOFile, InName);
    PutHead (GOFile);
    PutDate (NR, GOFile);
    PutInAddr (NR, GOFile);
    PutRefe (NR, GOFile);
    PutSalut (NR, GOFile);
    PUTPara1 (GOFile);
    GetAnswers (NR, Ansgate, Cal);
    IF NR.Gender = 'Male' THEN
        GetMaleLimits (NR, Cal)
    ELSE
        GetFemaleLimits (NR, Cal);
    PutEndLet (NR, GOFILE, Cal);
    MAKE_Qans_STR (NR);
    MakerEC (NR);
    MakeNAME (NR, keep);
    WorkSheet (NR, Cal);
END; /* IF */
END; /* NEW_LETTER */

******************************************************************************
PROCEDURE INITIALIZE (VAR Kal : CalcType);
(* Initializes the record type which holds the calculated values and the *)
(* array which holds the test answers and allows their manipulation. *)

(* RECEIVES - a VAR Calculated Values Record *)
(* RETURNS - an initialized Calculated Values Record and an initialized *)
(* test answer array *)
PROCEDURE GetDorL (VAR TR : PRectype);
(* This procedure prompts user to enter a D, if the client is a doctor, *)
(* or an L, if the client is a lawyer, and repeats until D or L is input.*)

(* RECEIVES - a VAR Working Record *)
(* RETURNS - a VAR Working Record with the DorL field set *)

(* MAIN_MENU-2, NEW_LETTER-2 Calls - N/A *)

PROCEDURE GetNames (VAR TR : PRectype; TotNames : integer;
    VAR Continuing : boolean; VAR EnName : string);
(* This procedure allows the user to specify the name of person tested. *)
(* The user is prompted to enter the name normally, First Middle Last, *)
(* Honorifics. The procedure is set to take up to a combination of five *)
(* names and honorifics. The entry is written to Pname, then manipulated*)
(* into Last First Middle, Honorific order and written to Rname. The *)
(* last name (up to 8 letters) becomes the Response Letter File name with*)
(* the first 2 initials and an "L" becoming the suffix. All of this is *)
(* displayed to the screen and the user is given a chance to reject it *)
(* and reenter the name. *)

(* RECEIVES - a VAR Working Record with Rnum value set, an integer value *)
(* of total records, a VAR boolean and a VAR string variable *)
(* RETURNS - a boolean value indicating if entry is to be continued, *)
(* a Working Record with the values of Rname and Pname set, *)
(* and a Response Letter File name *)

(* MAIN_MENU-2, NEW_LETTER-3 *)
(* MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-2 *)
(* Calls - GetChecks *)
(* MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-2 *)

P-9
PROCEDURE GetChecks (TR : PRectype; TotNames : integer;
   VAR Continuing : boolean);
(* This procedure checks the value of Rname against an array of the names*)
(* in the Names File. If found, it concatenates Rname and Rnum, alerts *)
(* the user that the name exists in the Working Records File and that the*)
(* current record will be filed as the concatenation, if entered. The *)
(* user is given the option of returning to the menu, against the chance *)
(* that a double entry is being made. *)
(* RECEIVES – a VAR Working Record with Rname and Rnum values set, an *)
(* integer value of total records, and a VAR boolean variable *)
(* RETURNS – a boolean value indicating if entry is to be continued and, *)
(* possibly, an altered Rname value *)
(* MAIN_MENU-2, NEW_LETTER-3, GetNames-1          / *)
(* MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-2, GetNames-1 / *)
(* MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-2, GetNames-1 / *)
(* Calls – N/A *)

PROCEDURE GetToday(VAR TR : PRectype);
(* This procedure writes the survey evaluation date to Working Records *)
(* File and provides dates for letters. The date is taken automatically *)
(* from the CPU registers, with no user interface. *)
(* RECEIVES – a VAR Working Record *)
(* RETURNS – a VAR Working Record with the Today field set *)
(* MAIN_MENU-2, NEW_LETTER-4 / MAIN_MENU-3, REC_LETTER-3 / *)
(* MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-3 / *)
(* MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-3 Calls – N/A *)

PROCEDURE GetInAdd (VAR TR : PRectype);
(* This procedure prompts the user for the Inside Address. It inserts a *)
(* ~ in the string each time the user hits carriage return, ending with *)
(* two consecutive carriage returns. The user may start over by entering*)
(* '999' on a separate line at any time. *)
(* RECEIVES – a VAR Working Record *)
(* RETURNS – a VAR Working Record with the CAddr field set *)
(* MAIN_MENU-2, NEW_LETTER-5 Calls – N/A *)
PROCEDURE GetGendr (VAR TR : PRectype);
(* This procedure prompts user to enter a 1 for male or a 2 for female *)
(* and repeats until 1 or 2 is input. *)

(* RECEIVES – a VAR Working Record
(* RETURNS – a VAR Working Record with the Gendr field set

(* MAIN_MENU-2, NEW_LETTER-6
(* MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-6
(* MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-6 Calls – N/A

PROCEDURE GetDates (VAR TR : PRectype);
(* This procedure prompts the user to enter, successively, the date of
(* birth, date of accident, and date of receipt in office. As each date
(* is entered, it is sent as a VAR parameter to ChekDate which either
(* validates it or returns an error message. If the date is invalid the
(* prompts repeat and the error message is displayed to the screen.

(* RECEIVES – a VAR Working Record
(* RETURNS – a VAR Working Record with Bdate, Adate, and Rdate fields set*)

(* MAIN_MENU-2, NEW_LETTER-7 Calls – ChekDate

PROCEDURE ChekDate (VAR date, why : string; VAR done : boolean);
(* This procedure screens dates for alphabetical entries or for numbers
(* which are out of range (e.g. the 13th month, 022995) as well as adding
(* slash marks in the correct format.

(* RECEIVES – a VAR string containing 6 integers representing a date
(* a VAR boolean for returning approval/disapproval of date
(* a VAR string for returning the rational for disapproval

(* RETURNS – a string containing a date in MM/DD/YY format
(* a boolean indicating the status of the date
(* a string containing, if applicable, the failure rational

(* MAIN_MENU-2, NEW_LETTER-7, GetDate-1
(* MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-7, NORM_BIRTHDAY
(* MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-8, NORM_TEST_DATE
(* MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-7, NORM_BIRTHDAY
(* MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-8, NORM_TEST_DATE
(* Calls – N/A

P-11
PROCEDURE GetTrauma (VAR TR : PRectype);
(* This procedure prompts for Y or N answers regarding documentation of *)
(* either head or psychological trauma and repeats prompts until a Y or *)
(* N is entered regarding each.*)

(* RECEIVES - a VAR Working Record *)
(* RETURNS - a VAR Working Record with HT and PT fields set *)

(* MAIN_MENU-2, NEW_LETTER-8        Calls - N/A *)

PROCEDURE GetAccid (VAR TR : PRectype);
(* This procedure prompts the user Y or N regarding the existence of an *)
(* accident description, repeating until Y or N is input. If N, "Not on *)
(* report." is set to ADisc. If Y, the user is prompted to enter the *)
(* description, which is read as ADisc.*)

(* RECEIVES - a VAR Working Record *)
(* RETURNS - a VAR Working Record with the ADisc field set *)

(* MAIN_MENU-2, NEW_LETTER-9        Calls - N/A *)

PROCEDURE GetSalut (VAR TR : PRectype);
(* This procedure prompts the user for the client's name for the Response*)
(* Letter salutation and reads the response as NSal.*)

(* RECEIVES - a VAR Working Record *)
(* RETURNS - a VAR Working Record with the NSal field set *)

(* MAIN_MENU-2, NEW_LETTER-10       Calls - N/A *)

PROCEDURE GetAdSym (VAR TR : PRectype);
(* This procedure prompts the user Y or N regarding the existence of any *)
(* reported additional symptoms, repeating until Y or N is input. If N, *)
(* "no other symptoms" is set to AdSymp. If Y, the user is prompted to *)
(* enter them, and is shown the first and last of the sentence they will *)
(* cause in the Response Letter. The user's response is read as AdSymp.*)

(* RECEIVES - a VAR Working Record *)
(* RETURNS - a VAR Working Record with the ADisc field set *)

(* MAIN_MENU-2, NEW_LETTER-11       Calls - N/A *)

P-12
PROCEDURE GetRspsFile (VAR GOFile : text; VAR EnName : string);
(* This procedure assigns individually named Response Letter files and, *)
(* by a different paths, assigns user specified norming files. *)

(* RECEIVES - a VAR text variable and a string holding a file name *)
(* RETURNS - a file variable assigned to the specified file name *)

(* MAIN_MENU-2, NEW_LETTER-12 / MAIN_MENU-3, REC_LETTER-4 /
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-4 /
(* MAIN_MENU-5, NORM_MENU-1.3 / MAIN_MENU-5, NORM_MENU-4.4 Calls - N/A *)

PROCEDURE PutHead (VAR GOFile : text);
(* This procedure writes the company's letter head (a program constant) *)
(* to the Response Letter file. *)

(* RECEIVES - a VAR text variable assigned to the Response Letter File *)
(* RETURNS - an updated Response Letter File *)

(* MAIN_MENU-2, NEW_LETTER-13 / MAIN_MENU-3, REC_LETTER-5 /
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-5 Calls - N/A *)

PROCEDURE PutDate (VAR TR : PRectype; VAR GOFile : text);
(* This procedure writes four blank lines, today's date, and four blank *)
(* lines to the Response Letter file. *)

(* RECEIVES - the current Working Record and a VAR text variable assigned*)
(* to the Response Letter File *)
(* RETURNS - an updated Response Letter File *)

(* MAIN_MENU-2, NEW_LETTER-14 / MAIN_MENU-3, REC_LETTER-6 /
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-6 Calls - N/A *)

PROCEDURE PutInAdd (VAR TR : PRectype; VAR GOFile : text);
(* This procedure writes inside address and a blank line to the Response *)
(* Letter file. *)

(* RECEIVES - the current Working Record and a VAR text variable assigned*)
(* to the Response Letter File *)
(* RETURNS - an updated Response Letter File *)

(* MAIN_MENU-2, NEW_LETTER-15 / MAIN_MENU-3, REC_LETTER-7 /
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-7 Calls - N/A *)

P-13
PROCEDURE PutRef (VAR TR : PRectype; VAR GOFile : text);
(* This procedure writes the tested person's Name, Gender, and Date of *)
(* Birth, as well as Date of Accident and Accident Description to the *)
(* Response Letter file in a reference format. *)

(* RECEIVES - the current Working Record and a VAR text variable assigned*)
(*     to the Response Letter File *)
(* RETURNS - an updated Response Letter File *)

(* MAIN_MENU-2, NEW_LETTER-16 / MAIN_MENU-3, REC.LETTER-8 / *
(* MAIN_MENU-4, RECS_MENU-1.1, REC.LETTER-8 Calls - N/A *)

*****************************************************************************
PROCEDURE PutSalut (VAR TR : PRectype; VAR GOFile : text);
(* This procedure writes two blank lines, the Salutation, the Doctor's or*)
(* Lawyer's name, colons, and a blank line to the Response Letter file. *)

(* RECEIVES - the current Working Record and a VAR text variable assigned*)
(*     to the Response Letter File *)
(* RETURNS - an updated Response Letter File *)

(* MAIN_MENU-2, NEW_LETTER-17 / MAIN_MENU-3, REC.LETTER-9 / *
(* MAIN_MENU-4, RECS_MENU-1.1, REC.LETTER-9 Calls - N/A *)

*****************************************************************************
PROCEDURE PutPara (VAR GOFile : text);
(* This procedure writes the first paragraph and the lead line to the *)
(* following 3 paragraphs to the Response Letter file. *)

(* RECEIVES - a VAR text variable assigned to the Response Letter File *)
(* RETURNS - an updated Response Letter File *)

(* MAIN_MENU-2, NEW_LETTER-18 / MAIN_MENU-3, REC.LETTER-10 / *
(* MAIN_MENU-4, RECS_MENU-1.1, REC.LETTER-10 Calls - N/A *)

*****************************************************************************
PROCEDURE GetAnswers (VAR TR : PRectype; AnsGate : integer;
        VAR Kal : Calctype);
(* This procedure receives user input regarding desired method for input *)
(* of answers and, after calling DISPLAY, calls GetAnsFast or GetAnsSlow *)
(* On return, the user is asked to accept displayed answers, make limited*)
(* corrections or delete all answers and start over. If corrections are *)
(* made, Procedures CONVERT and DISPLAY are called to update the screen. *)
(* If all answers are to be discarded, INITIALIZE, DISPLAY and GetAnsFast*)
(* or GetAnsSlow are called. Once answers are accepted, TALLY is called *)
(* to calculate weighted category scores, DISPLAY is called, and a "quick")
(* and dirty" categorical analysis is written to the screen. GetAnsSlow *)
(* is only accessible when drafting a new letter. The use of flags also *)
(* prevents any initial answer input when reading an existing record. *)

P-14
VAR
  sf : char; (* User inputs: F-answer fast or S-answer slow *)
  c : char; (* User inputs: Y-accept, N-correct, S-dump answers *)
  w, i : integer; (* Counter used for formatting screen/skipping lines *)
BEGIN
  clrscr;
  c := '0';
  FOR w := 1 TO 6 DO WRITELN;
  IF Ansgate = 2 THEN (* Choice only provided for new letter drafting *)
    REPEAT
      WRITELN ('Enter numbers one by one, checking answer each entry ' +
        ' - enter S');
      WRITELN;
      WRITELN (' OR ');
      WRITELN;
      WRITELN ('Enter all numbers, then check and correct as required ' +
        ' - enter F');
      READLN (sf);
      UNTIL (sf = 'S') OR (sf = 's') OR (sf = 'F') OR (sf = 'f')
    ELSE sf := 'F';
    clrscr;
    DISPLAY (TR);
    IF (sf = 'S') OR (sf = 's') THEN
      GetAnsSlow (TR)
    ELSE (sf = 'F') OR (sf = 'f') THEN
      GetAnsFast (TR, Ansgate);
    REPEAT
      WRITELN;
      WRITELN ('Happy with numbers? Y or y accepts them ');
      WRITELN (' N or n allows corrections.');
      WRITELN (' D or d dumps all numbers & restarts.');
    READLN (c);
IF ((c = 'n') OR (c = 'N')) THEN
    BEGIN
        WRITELN;
        WRITELN ('What is the number of question you wish to correct?');
        READLN (i);
        WRITELN;
        WRITELN ('What number should be entered for question ', i, '?');
        READLN (Qans[i].wt);
        CONVERT (i);
        DISPLAY (TR)
    END; (* IF *)
IF ((c = 'd') OR (c = 'D')) THEN
    BEGIN
        INITIALIZE (Cal);
        DISPLAY (TR);
        IF ((sf = 'S') OR (sf = 's')) THEN GetAnsSlow (TR)
          ELSE GetAnsFast (TR, AnsGate);
    END; (* IF *)
UNTIL ((c = 'y') OR (c = 'Y'));
TALLY (TR, kal);
DISPLAY (TR);
WRITELN;
WRITELN;
WITH kal DO BEGIN
              ' CACUM HAD ', EX, ' HITS');
    (* "CACUM" supplied by company; stands for raw exaggeration *)
    WRITELN;
    WRITELN ('FOR OTHER HITS:  D = ', D1:2:0, ' A = ', A1:2:0,
              ' AND P = ', P1:2:0);
    WRITELN;
    WRITELN ('& PERCENT HITS:  D = ', DP:2:0, ' A = ', AP:2:0,
              ' P = ', PP:2:0, '" AND E = ', EXP:2:0, '". ');
    END; (* WITH *)
END; (* GetAnswers *)

***************************************************************************************

PROCEDURE DISPLAY (JR :原理型);
(* This procedure writes the test questions' numbers, answers, and first *)
(* letters of the noun equivalent answers to the screen. Display is in *)
(* ordinal columns and includes the name of the survey participant. *)

(* RECEIVES - the current Working Record *)
(* RETURNS - a display to the screen *)

(* MAIN_MENU-2, NEW_LETTER-19, GetAnswers-1&5&7&11 | GetAnswers-2&8, *)
(* GetAnsSlow-2 | GetAnswers-3&9, GetAnsFast-3 / *)
(* MAIN_MENU-3, REC_LETTER-11, GetAnswers-1&5&7&11 | GetAnswers-3&9, *)
(* GetAnsFast-3 / *)
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-11, GetAnswers-1&5&7&11 | *)
(* GetAnswers-3&9, GetAnsFast-3 / *)

P-16
(* MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-4, GetAnswers-1&5&7&11 | *)
(* MAIN_MENU-5, NORM_MENU-4.5, GetAnswers-1&5&7&11 | *)
(* MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-4, *)
(* GetAnswers-1&5&7&11 | GetAnswers-3&9, GetAnsFast-3 *)
(* Calls – N/A *)

*****************************************************************************************/
PROCEDURE GetAnsSlow (VAR SR : PRectype);
(* This procedure allows users to enter answers one at a time and observe*)
(* the results after each entry. It was requested by the company. It is*)
(* not envisioned that it will be heavily used, as entering the answers *)
(* by this method is laborious, relative to its alternative – GetAnsFast.*)
(* A filter prevents entries which are nonnumeric or out of range. *)

(* RECEIVES – the current Working Record *)
(* RETURNS – all answers to the Qans array *)

(* MAIN_MENU-2, NEW_LETTER-19, GetAnswers-2&8 *)
(* Calls – DISPLAY and CONVERT *)

*****************************************************************************************/
PROCEDURE DISPLAY (JR : PRectype);

**************************************************************
PROCEDURE CONVERT (b : integer);
(* This procedure converts '1' to the first letter of the given answer: *)
(* 1 = never, 2 = occasionally, 3 = frequently, and 4 = constantly. This*)
(* is used for screen and WorkSheet display as a memory aide and is used *)
(* by the program to screen against illegal entries. *)

(* RECEIVES – an integer representing an array index *)
(* RETURNS – an updated answer array *)

(* MAIN_MENU-2, NEW_LETTER-19, GetAnswers-4 | GetAnswers-2&8, GetAnsSlow-1 | GetAnswers-3&9, GetAnsFast-1&2 *)
(* MAIN_MENU-3, REC_LETTER-11, GetAnswers-4 | GetAnswers-3&9, GetAnsFast-1&2 *)
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-11, GetAnswers-4 | GetAnswers-3&9, GetAnsFast-1&2 *)
(* MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-4, *)
(* GetAnswers-4 | GetAnswers-3&9, GetAnsFast-1&2 *)
(* MAIN_MENU-5, NORM_MENU-4.5, GetAnswers-4 | GetAnswers-3&9, GetAnsFast-1&2 *)
(* MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-4, *)
(* GetAnswers-4 | GetAnswers-3&9, GetAnsFast-1&2 *)
(* Calls – N/A *)

P-17
PROCEDURE GetAnsFast (VAR SR : PRectype; SS : integer);
(* This procedure allows a user to enter answers as one 35 digit number. *)
(* Checks provide warnings to the user for incorrect, incomplete, or *)
(* excess (>35) entries and prompt for correction. *)

(* RECEIVES - the current Working Record, an integer flag which allows *)
(* complete Working Records to correctly reach CONVERT and DISPLAY *)
(* RETURNS - all answers to the Qans array *)

(* MAIN_MENU-2, NEW_LETTER-19, GetAnswers-3&9 / *)
(* MAIN_MENU-3, REC_LETTER-11, GetAnswers-3&9 / *)
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-11, GetAnswers-3&9 / *)
(* MAIN_MENU-5, NORM_Menu-1.4, BUILD_NORM_FILE-4, GetAnswers-3&9 / *)
(* MAIN_MENU-5, NORM_Menu-4.5, GetAnswers-3&9 / *)
(* MAIN_MENU-5, NORM_Menu-6.4, BUILD_NORM_FILE-4, GetAnswers-3&9 *)
(* Calls - DISPLAY and CONVERT *)

PROCEDURE DISPLAY (JR : PRectype);

AS SHOWN ABOVE

PROCEDURE CONVERT (b : integer);

AS SHOWN ABOVE

PROCEDURE INITIALIZE (VAR Kal : Calctype);

AS SHOWN ABOVE

PROCEDURE TALLY (VAR C : Calctype);
(* After all answers have been loaded to the answer array, this procedure*)
(* assigns a weight to each answer and, by category; totals the weighted *)
(* score, totals the answers other than "never", and calculates the per- *)
(* centage of the answers other than "never" to the number of questions *)
(* within that category. *)

(* RECEIVES - the current Calculated Values Record *)
(* RETURNS - the Calculate Values Record with all values set, save the *)
(* verbal descriptions for dstring, astring, pstring, and estring*)

(* MAIN_MENU-2, NEW_LETTER-19, GetAnswers-10 / *)
(* MAIN_MENU-3, REC_LETTER-11, GetAnswers-10 / *)
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-11, GetAnswers-10 / *)
(* MAIN_MENU-5, MN-2.2, DO_THE_STATS-6 / *)
(* NORM_MENU-4.5, GetAnswers-10 Calls - N/A *)

VAR

i : integer; (* Used as the array index *)
PLUG : real ; (* Questions are weighed according to frequency *)
BEGIN
FOR i := 1 to 35 DO
BEGIN
  WITH Qans[i] DO
  BEGIN
    IF wt = '1' THEN PLUG := 0 (* Never *)
    ELSE IF wt = '2' THEN PLUG := 1.0 (* Occasionally *)
    ELSE IF wt = '3' THEN PLUG := 2.0 (* Frequently *)
    ELSE IF wt = '4' THEN PLUG := 2.5 (* Constantly *)
    END;

    ('Program is broken; contact Tedford - (512) 993-5503.');

    IF (i = 4) OR (i = 5) OR (i = 6) OR (i = 9)
    OR (i = 12) OR (i = 16) OR (i = 17) OR (i = 18)
    OR (i = 19) OR (i = 22) OR (i = 24) OR (i = 27)
    OR (i = 30) OR (i = 31) OR (i = 34) OR (i = 35)
    THEN BEGIN
      C.D := C.D + PLUG; (* Weighted Depression *)
      IF PLUG > 0 THEN C.D1 := C.D1 + 1.0; (* Raw Depression *)
      END (* IF for D *);
    IF (i = 2) OR (i = 3) OR (i = 8) OR (i = 10)
    OR (i = 11) OR (i = 15) OR (i = 18) OR (i = 21)
    OR (i = 23) OR (i = 25) OR (i = 29) OR (i = 30)
    OR (i = 31) OR (i = 32) OR (i = 33) OR (i = 35)
    THEN BEGIN
      C.A := C.A + PLUG; (* Weighted Anxiety *)
      IF PLUG > 0 THEN C.A1 := C.A1 + 1.0; (* Raw Anxiety *)
      END (* IF for A *);
    IF (i = 1) OR (i = 2) OR (i = 3) OR (i = 7)
    OR (i = 12) OR (i = 13) OR (i = 14) OR (i = 18)
    OR (i = 20) OR (i = 21) OR (i = 24) OR (i = 26)
    OR (i = 27) OR (i = 28) OR (i = 31) OR (i = 32)
    THEN BEGIN
      C.P := C.P + PLUG; (* Weighted PTSD *)
      IF PLUG > 0 THEN C.P1 := C.P1 + 1.0; (* Raw PTSD *)
      END (* IF for P *);
    IF PLUG = 2.5 THEN C.EX1 := C.EX1 + 1.0; (* Real # to Get % *)
    IF PLUG = 2.5 THEN C.EX := C.EX + 1; (* Raw Exaggeration *)
    END (* FOR *);

  C.DP := C.D1 / 16.0 * 100; (* % of DEPRESSION hits *)
  C.AP := C.A1 / 16.0 * 100; (* % of ANXIETY hits *)
  C.PP := C.P1 / 16.0 * 100; (* % of POST TRAUMATIC STRESS hits *)
  C.EXP := C.EX1 / 35.0 * 100; (* % of EXAGGERATION hits *)
END; (* TALLY *)
PROCEDURE GetMaleLimits (VAR TR : PRectype; VAR kal : Calctype);

(* In this procedure limits, against which tested males are measured, are *)
(* set into the program and the value of teller, an accumulator of the *)
(* indicators of disorder, is determined. It and its female counterpart,*)
(* GetFemaleLimits, are the procedures which would be changed to accomplish*)
(* any renorming of the PLUGFORM system by the formulae: *)
(* up = 1.04 * standard deviation + mean  (All answers are *)
(* lo = .525 * standard deviation + mean  rounded up to the *)
(* neg = mean nearest .5 )*)
(* Standard deviations and means are taken from the Male Norming File. *)
(* It was not the company's desire to do this dynamically, within the *)
(* system. They are, therefore, static or "hard wired" in the procedure.*)
(* As the limits are set for each disorder, they and the weighted score *)
(* and unset string in their category are sent to GetWordLimits to set a *)
(* value into the string. *)

(* RECEIVES - the current Working Record and current Calculated Values *)
(* Record, both as VAR parameters *)
(* RETURNS - the Calculate Values Record with dstring, astring, pstring, *)
(* & estring values set and the Working Record with teller set *)

(* MAIN_MENU-2, NEW_LETTER-20 / MAIN_MENU-3, REC_LETTER-12 / *)
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-12 *)
(* Calls - GetWordLimits *)

'AR
   up , (* These are the respective threshold values for calling *)
   lo , (* the relative degrees to which a disorder is indicated.*)
   neg : real; (* Rounding up + using G.T. = erring towards caution. *)

BEGIN
   WITH kal DO BEGIN
   {$V-}  (* Disables VAR String-Checking for short string pass*)
   up := 18.0;  (* Upper limit for Dep, is 'high' if above up *)
   lo := 14.0;  (* Lower limit for Dep, is 'moderate' if above low *)
   neg := 12.0;  (* Neg to low = 'low', below neg is 'negligible' *)
   GetWordLimits (dstring, D, up, lo, neg);
   up := 17.0;  (* Upper limit for Anx, is 'high' if above up *)
   lo := 14.0;  (* Lower limit for ANX, is 'moderate' if above low *)
   neg := 11.0;  (* Neg to low = 'low', below neg is 'negligible' *)
   GetWordLimits (astring, A, up, lo, neg);
   up := 18.0;  (* Upper limit for PSTD, is 'high' if above up *)
   lo := 16.0;  (* Lower limit for PSTD, is 'moderate' if above low *)
   neg := 13.0;  (* Neg to low = 'low', below neg is 'negligible' *)
   GetWordLimits (pstring, P, up, lo, neg);
   up := 11.0;  (* Upper limit for Exg, is 'high' if above up *)
   lo := 8.0;  (* Lower limit for Exg, is 'moderate' if above low *)
   neg := 4.0;  (* Neg to low = 'low', below neg is 'negligible' *)

P-20
GetWordLimits (estring, EX, up, lo, neg);  
{SV+}  (* VAR String-Checking re-enabled *)  
WRITELN;  
WRITELN ('D is ', dstring, '. A is ', astring, '. P is ', pstring,  
'. E is ', estring, '.');  
WRITELN;  
WRITELN ('    scored ',  
TR.Today);  
WRITELN ('This is just a check.');  
IF dstring = 'high' THEN  
teller := teller + 3  
ELSE IF dstring = 'moderate' THEN  
teller := teller + 2;  
IF astring = 'high' THEN  
teller := teller + 3  
ELSE IF astring = 'moderate' THEN  
teller := teller + 2;  
IF pstring = 'high' THEN  
teller := teller + 3  
ELSE IF pstring = 'moderate' THEN  
teller := teller + 2;  
END;  
END;  
(* GetMaleLimits *)

**************************************************************************  
PROCEDURE GetFemaleLimits (VAR TR : PRectype; VAR kal : Calctype);  
(* This procedure is identical to GetMaleLimits, except that the Female *)  
(* Norming File was used to calculate up, lo, and neg. Consequently, *)  
(* they are of different values in this procedure. *)  

(* RECEIVES - the current Working Record and current Calculated Values *)  
(* Record, both as VAR parameters *)  
(* RETURNS - the Calculate Values Record with dstring, astring, pstring, *)  
(* & estring values set and the Working Record with teller set *)  

(* MAIN_MENU-2, NEW_LETTER-20 / MAIN_MENU-3, REC_LETTER-12 /  
*)  
(*MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-12  
*)  
(* Calls - GetWordLimits *)

P-21
PROCEDURE GetWordLimits (VAR gstring : string; Q, LimUp, LimLow, NegLim : real);

(* This procedure captures names of result levels, for use in WorkSheet *)
(* and Response Letter. *)

(* RECEIVES - a VAR string variable, a real number set to the weighted *)
(* value of answers within a category, and three real numbers *)
(* containing limits for degrees of probability of disorder *)
(* RETURNS - a string containing the (verbal) probability of disorder *)

(* MAIN_MENU-2, NEW_LETTER-20, GetMaleLimits-1&2&3&4  
(*) MAIN_MENU-2, NEW_LETTER-20, GetFemaleLimits-1&2&3&4  
(*) MAIN_MENU-3, REC_LETTER-12, GetMaleLimits-1&2&3&4  
(*) MAIN_MENU-3, REC_LETTER-12, GetFemaleLimits-1&2&3&4  
(*) MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-12, GetMaleLimits-1&2&3&4  
(*) MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-12, GetFemaleLimits-1&2&3&4  
(*) Calls - N/A

BEGIN
  IF Q <= NegLim THEN gstring := 'negligible'
  ELSE IF Q <= LimLow THEN gstring := 'low'
      ELSE IF (Q > LimLow) AND (Q <= LimUp) THEN gstring := 'moderate'
          ELSE gstring := 'high';
END; (* GetWordLimits *)

PROCEDURE PutEndLet (TR : PRectype; VAR GOFile : text; kal : CalcType);

(* This finishes the Response Letter. The procedure calls PutGenPara 3 *)
(* times - writing area specific paragraphs - and PutHowever 3 times. *)
(* A paragraph which covers additional symptoms, or the lack of, follows.*)
(* The next paragraph deals with symptom exaggeration, and the cutoffs *)
(* for exaggeration may be altered through the code in this procedure. *)
(* The procedure will also insert a WordPerfect macro trigger into the *)
(* Response Letter File, if the client is a doctor. That macro will, if *)
(* the trigger is present, search for and change profession specific *)
(* phraseology. *)

(* RECEIVES - the current Working Record, a VAR text variable assigned *)
(* to the Response Letter File, and the current Calculated *)
(* Values Record *)
(* RETURNS - an updated Response Letter File *)

(* MAIN_MENU-2, NEW_LETTER-21  
(*) MAIN_MENU-3, REC_LETTER-13  
(*) MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-13  
(*) Calls - PutHowever, PutGenPara *)

P-22
PROCEDURE PutHowever (BP : real; bstring : string; VAR BH : boolean);
(* This procedure is solely used to generate a 'however' in the general *)
(* paragraph – PROCEDURE GenPara – for those situations in which it *)
(* might appear to the layman that the test indicated something other *)
(* than what it did. e.g., >50% hits, but low probability of problems. *)

(* RECEIVES – a real variable set to the percentage of positive answers, *)
(* a string variable indicating the probability of disorder, *)
(* and a VAR boolean variable. *)
(* RETURNS – the boolean variable set to govern the use of "However". *)

(* MAIN_MENU-2, NEW_LETTER-21, PutEndLet-1&3&5
MAIN_MENU-3, REC_LETTER-13, PutEndLet-1&3&5
MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-13, PutEndLet-1&3&5
(*
Calls – N/A *)

PROCEDURE PutGenPara (VAR GOFile : text; sy, sp : real; st : string;
BH : boolean);
(* This procedure is called 3 times from PutEndLet, for paragraphs on *)
(* Depression, Anxiety and Post Traumatic Stress Syndrome. The procedure*)
(* adds disorder specific raw scores and percentages to the paragraphs *)
(* and, based on the boolean variable, may insert the word "However". *)

(* RECEIVES – a VAR text variable assigned to the Response Letter File, *)
(* two real numbers representing raw score and percentage of *)
(* positive answers for a category, a string containing the *)
(* probability of the disorder, and a boolean variable to *)
(* govern the use of "However". *)
(* RETURNS – an updated Response Letter File *)

(* MAIN_MENU-2, NEW_LETTER-21, PutEndLet-2
MAIN_MENU-3, REC_LETTER-13, PutEndLet-2
MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-13, PutEndLet-2&4&6
(*
Calls – N/A *)

PROCEDURE MAKE_Qans_STRING (VAR TR : PRectype);
(* This procedure converts the answer array into a 35 number string for *)
(* storage in the Working Records file. *)

(* RECEIVES – the current Working Record as a VAR parameter *)
(* RETURNS – a completed (Qans now set) Working Record *)

(* MAIN_MENU-2, NEW_LETTER-22
MAIN_MENU-4, RECS_MENU-1.5
MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-5
MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-5
(*
Calls – N/A *)
PROCEDURE MakeRec (TR : PRectype);
.* This procedure appends a record to the Working Records file. *

(* RECEIVES – the current Working Record *
(* RETURNS – an updated Working Records File *

(* MAIN_MENU-2, NEW_LETTER-23 / MAIN_MENU-4, RECS_MENU-1.6 Calls - N/A *)

PROCEDURE MakeName (TR : PRectype; VAR TotNames : integer);
(* This procedure appends a name to the Name File and updates the number *)
(* of names contained therein. *

(* RECEIVES – the current Working Record and a VAR integer containing *)
(* the number of records in the Working Records File *
(* RETURNS – an updated Name File *

(* MAIN_MENU-2, NEW_LETTER-24 / MAIN_MENU-4, RECS_MENU-1.7 *
(* Calls – LOAD_NAME_ARRAY *

PROCEDURE LOAD_NAME_ARRAY (TotNames : integer);
(* This procedure loads the array 'Nam' from the Names.rec, a file which *)
(* contains all (and only) the names from the Working Records file, in *)
(* Last First Middle order. *

* RECEIVES – an integer representing total names *
(* RETURNS – a loaded name array *

(* MAIN-2 *
(* MAIN_MENU-2, NEW_LETTER-24, MakeName-1 *
(* MAIN_MENU-4, RECS_MENU-3.8 *
(* MAIN_MENU-4, RECS_MENU-4.1 Calls - N/A *)

PROCEDURE WORKSHEET (TR : PRectype; kal : Calctype);
(* This procedure prints out a worksheet which contains all information *)
(* input by the user, the tested person's answers, interpretive scores, *)
(* and the actions to be taken by the clerical personnel. It is here *)
(* a "permanent" integer, provided by the company resides. If exceeded *)
(* by values accumulated within a program variable, it indicates possible *)
(* psychological problems. This value may not be changed externally. *)

(* RECEIVES – a Working Record and a Calculated Values record, both with *)
(* all fields set. *
(* RETURNS – causes the Work Sheet to printed *

(* MAIN_MENU-2, NEW_LETTER-25 *
(* MAIN_MENU-3, REC_LETTER-14 *
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-14 *
(* Calls – BIGIF, SET_SPACE, GetClientName *

P-24
PROCEDURE BIGIF (t : integer; FAKE : boolean; JR : PRectype;
VAR m : string; GOFile : text);
(* This procedure determines which WordPerfect macro will complete the *)
(* Response Letter and inserts a trigger for the macro into the Response *)
(* Letter File. If t >= 3 it is an indication of emotional trauma - a *)
(* "no" in PASS TOO LOW section of the Work Sheet. If FAKE = TRUE, it *)
(* indicates exaggeration and a "yes" to PASS TOO HIGH on the Work Sheet.*)
(* PT is reported emotional trauma from report and checked on Work Sheet.*)
(* HT is reported head trauma from report and checked on Work Sheet. *)
(* Each of the eight macros results in a different Response Letter and *)
(* the also govern which actions are marked on the Work Sheet to instruct*)
(* clerical personnel as to what to do with the finished product. *)
(* RECEIVES - an integer holding the value of (the indicator accumulator)*)
(* teller, a boolean variable indicating if estring >= moderate, *)
(* the current Working Record, a VAR string variable to return *)
(* the macro to WORKSHEET, and the Response Letter file variable *)
(* RETURNS - complete Response Letter File and the macro to WORKSHEET *)
(* MAIN_MENU-2, NEW_LETTER-25, WORKSHEET-1 *)
(* MAIN_MENU-3, REC_LETTER-14, WORKSHEET-1 *)
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-14, WORKSHEET-1 Calls - N/A *)

BEGIN
  WITH JR DO
    IF
      (t >= 3) AND (PT = 'N') AND (HT = 'N') AND FAKE THEN m := 'CCCC'
    ELSE IF
      (t >= 3) AND (PT = 'N') AND (HT = 'Y') AND FAKE THEN m := 'GGGG'
    ELSE IF
      (t < 3) AND (PT = 'N') AND (HT = 'N') THEN m := 'DDDD'
    ELSE IF
      (t < 3) AND (PT = 'N') AND (HT = 'Y') THEN m := 'HHHH'
    ELSE IF
      (t < 3) AND (PT = 'Y') AND (HT = 'N') THEN m := 'BBBB'
    ELSE IF
      (t < 3) AND (PT = 'Y') AND (HT = 'Y') THEN m := 'FFFF'
    ELSE IF
      (t >= 3) AND (HT = 'N') THEN m := 'AAAA'
    ELSE IF
      (t >= 3) AND (HT = 'Y') THEN m := 'EEEE'
    ELSE m := 'No Fit'; (* No Fit is fail safe; can't logically happen.*)
    APPEND (GOfile);
    WRITELN (GOfile, m);
    CLOSE (GOfile);
END; (* BIGIF *)
PROCEDURE SET SPACE (VAR norman : string);
(* This procedure is solely for formatting dstring, astring, pstring, and*)
(* estring values ('negligible' through 'high') to take up equal space *)
(* one the Work Sheet.
(* RECEIVES - a string holding a 3 to 10 letter word
(* RETURNS - a string holding a 3 to 10 letter word, padded with spaces *)
(* MAIN_MENU-2, NEW_LETTER-25, WORKSHEET-2&3&4&5 /
(* MAIN_MENU-3, REC_LETTER-14, WORKSHEET-2&3&4&5 /
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-14, WORKSHEET-2&3&4&5 *
(* Calls - N/A *

PROCEDURE GetClientName (TR : PRectype; VAR CLI_NAME : string);
(* This procedure extracts the client's name from the inside address by *)
(* copying the 'top line' of the address.
(* RECEIVES - the current Working Record and a VAR string variable
(* RETURNS - the client's name in the string variable
(* MAIN_MENU-2, NEW_LETTER-25, WORKSHEET-6 /
(* MAIN_MENU-3, REC_LETTER-14, WORKSHEET-6 /
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-14, WORKSHEET-6 Calls - N/A *)

PROCEDURE REC_LETTER (VAR SR : PRectype; VAR Cal : Calctype);
(* This procedure calls the procedures required to retrieve survey scores*)
(* and to reconstruct Response Letters and WorkSheets from the Working *)
(* Records file. It is also used for correction of stored test scores. *)
(* RECEIVES - a VAR Working Record and a VAR Calculated Values Record *
(* RETURNS - a completed Response Letter File,
(* a Working Record with all fields set,
(* a Calculated Values Record with all fields set,
(* a Work Sheet, and (perhaps)
(* an updated Working Record and/or Working Records File.
(* Calls - INITIALIZE, READ_WORKING_RECORD, GetToday, GET_Qans_ARRAY, *
(* GetRspsFile, PutHead, PutDate, PutInAdd, PutREfer, PutSalut, PutPara1,*)
(* GetAnswers, GetMaleLimits or GetFemeLimits, PutEndLet, and WorkSheet *

P-26
"VAR
  AnsGate : integer; (* A flag for speed option in PROCEDURE GetAnswers *)
  InName : string ; (* Used to get return of Response Letter file *)
  bail   : char ; (* "just in case" against unanticipated sex problem*)
BEGIN
  INITIALIZE (Cal);
  clrsr;
  AnsGate := 3;
  InName := ";
  READ_WORKING_RECORD (SR, InName);
  IF InName <> 'ZZZZ' THEN
    BEGIN
      GetToday (SR);
      GET_Qans_ARRAY (SR);
      GetRspsFile (GOFile, InName);
      PutHead (GOFile);
      PutDate (SR, GOFile);
      PutInAdd (SR, GOFile);
      PutREfer (SR, GOFile);
      PutSalut (SR, GOFile);
      PutPara1 (GOFile);
      GetAnswers (SR, AnsGate, Cal);
      IF SR.Gender = 'Male' THEN
        GetMaleLimits (SR, Cal)
      ELSE IF SR.Gender = 'Female' THEN
        GetFemaleLimits (SR, Cal)
      ELSE (* This is omitted for clarity; see code listings. *)
        PutEndLet (SR, GOFILE, Cal);
      WorkSheet (SR, Cal)
    END (* IF *);
  END; (* REC_LETTER *)
END; (* REC_LETTER *)

****************************************************************************

PROCEDURE INITIALIZE (VAR Mal : Calctype);
  AS SHOWN ABOVE

****************************************************************************

PROCEDURE READ_WORKING_RECORD (VAR TR : Prectype; VAR InCome : string);
(* This procedure allows the user to retrieve a record from the Working *)
(* Records file in order to either recreate a Response Letter file or to *)
(* compare the indicated person's survey results against normed data. *)

(* RECEIVES - a VAR Working Record and a VAR string which may be coded *)
(* RETURNS - a loaded Working Record, a Response Letter file name, (if *)
(* required) or a Working Record set to return to calling menu *)

(* MAIN_MENU-3, REC_LETTER-2
(* MAIN_MENU-4, RECS_MENU-1.1, REC_LETTER-2
(* MAIN_MENU-4, RECS_MENU-2.1
(* MAIN_MENU-5, NORM_MENU-4.2
(* Calls - CoverDups, ConvertName

P-27
This procedure prompts the user for the name of the person with whose record he wishes work. If the name entered is not in the Working Records File, the user is informed, advised how to proceed, and returned to the Main Menu. If the name is valid, the record is loaded. If the user is either reconstructing a letter or correcting a record, PROCEDURE ConvertName is called to build a file name.

**************************************************************************
PROCEDURE CoverDups (VAR TR : PRectype);
(* This procedure accesses the name alphabetically following the name the *)
(* user has supplied to the program for letter reconstruction, record *)
(* correction, record deletion, or record comparison. Should the name *)
(* provided be included in that name, the user is shown both names and *)
(* prompted to either continue or return to the calling menu. *)
(* MAIN MENU-3, REC LETTER-2, READ WORKING RECORD-1 / *)
(* MAIN MENU-4, RECS MENU-1.1, REC LETTER-2, READ WORKING RECORD-1 / *)
(* MAIN MENU-4, RECS MENU-2.1, READ WORKING RECORD-1 / *)
(* MAIN MENU-5, NORM MENU-4.2, READ WORKING RECORD-1 *)
(* Calls - N/A *)

**************************************************************************
PROCEDURE ConvertName (VAR TR : PRectype; VAR Fill : string);
(* This procedure converts a user provided name, confirmed to be in the *)
(* Working Records file, into a Response Letter file name, so that the *)
(* Response Letters may be reconstructed from records in that file. *)
(* RECEIVES - Working Record with all values set and a VAR string. *)
(* RETURNS - Response Letter file name. *)
(* MAIN MENU-3, REC LETTER-2, READ WORKING RECORD-1 / *)
(* MAIN MENU-4, RECS MENU-1.1, REC LETTER-2, READ WORKING RECORD-1 *)
(* Calls - N/A *)

**************************************************************************
PROCEDURE GetToday(VAR TR : PRectype);

**************************************************************************
PROCEDURE GET Qans ARRAY (TR : PRectype);
(* This procedure loads an answer array from a (stored condition) string.*)
(* RECEIVES - the current Working Record holding Qans *)
(* RETURNS - an answer array, loaded from the string Qans *)
(* MAIN MENU-3, REC LETTER-3 / *)
(* MAIN MENU-4, RECS MENU-1.1, REC LETTER-3 / *)
(* MAIN MENU-5, NORM MENU-2.2, DO THE STATS-5 / *)
(* MAIN MENU-5, NORM MENU-4.3 Calls - N/A *)
PROCEDURE GetRspsFile (VAR GOFile : text; VAR EnName : string);

PROCEDURE PutHead (VAR GOFile : text);

PROCEDURE PutDate (VAR TR : PRectype; VAR GOFile : text);

PROCEDURE PutInAdd (VAR TR : PRectype; VAR GOFile : text);

PROCEDURE PutREfer (VAR TR : PRectype; VAR GOFile : text);

PROCEDURE PutSalut (VAR TR : PRectype; VAR GOFile : text);

PROCEDURE PutPara1 (VAR GOFile : text);

PROCEDURE GetAnswers (VAR TR : PRectype; AnsGate : integer;
VAR Kal : Calctype);

PROCEDURE DISPLAY (JR : PRectype);

PROCEDURE GetAnsSlow (VAR SR : PRectype);

PROCEDURE GetAnsFast (VAR SR : PRectype; SS : integer);

PROCEDURE CONVERT (b : integer);

PROCEDURE TALLY (VAR C : Calctype);

PROCEDURE GetMaleLimits (VAR TR : PRectype; VAR kal : Calctype);

PROCEDURE GetFemaleLimits (VAR TR : PRectype; VAR kal : Calctype);

PROCEDURE GetWordLimits (VAR gstring : string; Q, LimUp, LimLow, NegLim : 
real);

P-29
PROCEDURE PutEndLet (TR : PRectype; VAR GOFile : text; kal : Calctype);

(* AS SHOWN ABOVE *)
PROCEDURE PutHowever (BP : real; bstring : string; VAR BH : boolean);

(* AS SHOWN ABOVE *)
PROCEDURE PutGenPara (VAR GOFile : text; sy, sp : real; st : string;
BH : boolean);

(* AS SHOWN ABOVE *)
PROCEDURE WORKSHEET (TR : PRectype; kal : Calctype);

(* AS SHOWN ABOVE *)
PROCEDURE BIGIF (t : integer; FAKE : boolean; JR : PRectype;
VAR m : string; GOFile : text);

(* AS SHOWN ABOVE *)
PROCEDURE SET_SPACE (VAR norman : string);

(* AS SHOWN ABOVE *)
PROCEDURE GetClientName (TR : PRectype; VAR CLI_NAME : string);

(* AS SHOWN ABOVE *)
(* MAIN_MENU OPTION 4 Visit RECORD MANIPULATION MENU. *)
(* MAIN_MENU-4 *)

VAR
RecSet : char ; (* Used to accept user's pick from this menu *)
WorkBy : string ; (* For READ_WORKING_RECORD, blocks ConvertName*)
WhichKey : integer; (* Used to accept field to sort records on *)

BEGIN
REPEAT
  clrscr;
  WRITELN;
  WRITELN (' RECORD MANIPULATION MENU');
  WRITELN;
  WRITELN (' ENTER If You Wish To '-');
  WRITELN (' 1 Correct survey answer(s). (This will take a '+
         ' few minutes.)');
  WRITELN (' 2 Delete a complete record. (This will take a '+
         ' few minutes.)');
  WRITELN (' 3 Sort records on a choice of keys. ');
  WRITELN (' 4 Obtain alphabetized listing of pertinent data.');
  WRITELN (' 5 Seek help on this menu. ');
  WRITELN (' 6 Return to the MAIN MENU. ');

  P-30
WRITELN;
WRITE ('   ');
READLN (RecSet);
crlscr;
IF RecSet = '1' THEN
BEGIN
  REC_LETTER (RR, Cal);
  IF RR.Rname <> 'ZZZZ' THEN
  BEGIN
    REMOVE_RECORD (RR);
    MAKE_Qans_STRING (RR);
    MAKEREc (RR);
    END; (* IF *)
  END; (* IF *);
IF RecSet = '2' THEN
BEGIN
  Workby := 'NN';
  READ_WORKING_RECORD (RR, Workby);
  IF RR.Rname <> 'ZZZZ' THEN
  BEGIN
    REMOVE_NAME (RR, TotNames);
    REWRITE_NameFile (TotNames);
    REMOVE_RECORD (RR);
    END; (* IF *)
  END; (* IF *);
IF RecSet = '3' THEN
BEGIN
  SAY_WhichKey (WhichKey);
  LOAD_ANY_ARRAY (TotNames, WhichKey);
  IF ((WhichKey >= 8) AND (WhichKey <= 10)) THEN
  END_FOR_END (TotNames);
  SHELLSORT (TotNames);
  IF ((WhichKey >= 8) AND (WhichKey <= 10)) THEN
  BEGIN
    END_FOR_END (TotNames);
    END_FOR_END (TotNames); /* Yes, twice to restore. */
    END; (* IF *)
  GET_ANY_ORDER_INFO (RR, TotNames, WhichKey);
  LOAD_NAME_ARRAY (TotNames);
  END; (* IF *)
IF RecSet = '4' THEN
BEGIN
  LOAD_NAME_ARRAY (TotNames);
  SHELLSORT (keep);
  REWRITE_NameFile (TotNames);
  GET_PERMA_ALPHA_INFO (RR);
  END; (* IF *)
IF RecSet = '5' THEN ReadInst (8);
  UNTIL RecSet = '6';
END; (* RECS_MENU *)
PROCEDURE GetAnsSlow (VAR SR : PRectype);

 PROCEDURE GetAnsFast (VAR SR : PRectype; SS : integer);

 PROCEDURE CONVERT (b : integer);

 PROCEDURE TALLY (VAR C : Calctype);

 PROCEDURE GetMaleLimits (VAR TR : PRectype; VAR kal : Calctype);

 PROCEDURE GetFemaleLimits (VAR TR : PRectype; VAR kal : Calctype);

 PROCEDURE GetWordLimits (VAR gstring : string; Q, LimUp, LimLow, NegLim : real);

 PROCEDURE PutEndLet (TR : PRectype; VAR GOFile : text; kal : Calctype);

 PROCEDURE PutHowever (BP : real; bstring : string; VAR BH : boolean);

 PROCEDURE PutGenPara (VAR GOfile : text; sy, sp : real; st : string;
 DH : boolean);

 PROCEDURE WORKSHEET (TR : PRectype; kal : Calctype);

 PROCEDURE BIGIF (t : integer; FAKE : boolean; JR : PRectype;
 VAR m : string; GOFile : text);

 PROCEDURE SET_SPACE (VAR norman : string);

 PROCEDURE GetClientName (TR : PRectype; VAR CLI_NAME : string);
PROCEDURE REMOVE_RECORD (PR : PRectype);
(* This procedure removes a record from the Working Records File by the *)
(* expedient of writing every record other than the desired deletion to *)
(* a new file and then copying that file to the Working Records File. *)
(* RECEIVES - the current Working Record *)
(* RETURNS - the Working Records File with the old copy of the current *)
(* record removed *)
(* MAIN_MENU-4, RECS_MENU-1.4 / MAIN_MENU-4, RECS_MENU-2.3 *)
(******************************************************************************)
PROCEDURE MAKE_Qans_STRING (VAR TR : PRectype);

(******************************************************************************)
PROCEDURE MakeRec (TR : PRectype);  

(******************************************************************************)
(* MAIN_MENU OPTION 4  Visit RECORD MANIPULATION MENU *)
(******************************************************************************)
(* RECS_MENU OPTION 2  Delete a complete record. *)
(******************************************************************************)
PROCEDURE READ_WORKING_RECORD (VAR TR : PRectype; VAR InComE : string);

(******************************************************************************)
PROCEDURE CoverDups (VAR TR : PRectype);

(******************************************************************************)
PROCEDURE REMOVE_NAME (PR : PRectype; VAR Totnames : integer);
(* This procedure removes a name from an array of names by comparisons *)
(* to the user provided name and index manipulation. *)
(* RECEIVES - a Working Record containing the name to be deleted and an *)
(* integer containing the total number of names *)
(* RETURNS - an updated name array and an updated total number of names *)
(* MAIN_MENU-4, RECS_MENU-1.2 *)

(******************************************************************************)
PROCEDURE REWRITE_NAMEFILE (TotNames : integer);
(* This procedure rewrites the NAMES.REC file to conform to the array of *)
(* names. *)
(* RECEIVES - an integer containing the total number of names *)
(* RETURNS - an updated NAMES.REC file *)

(* MAIN-4 *)
(* MAIN_MENU-4, RECS_MENU-2.2 *)
(* MAIN_MENU-4, RECS_MENU-4.3 *)

Calls - N/A *)

(******************************************************************************)

P-34
**PROCEDURE REMOVE_RECORD (PR : PRectype);**  

**AS SHOWN ABOVE**

(* MAIN_MENU OPTION 4 Visit RECORD MANIPULATION MENU. *)

(* RECS_MENU OPTION 3 Sort records on a choice of keys. *)

**PROCEDURE SAY_WhichKey (VAR WhichKey : integer);**

(* This procedure prompts users as to which key - Date of Birth, Date of *)

(* of Accident, or Date of Receipt - they wish to use for sorting the *)

(* Working Records File. *)

(* RECEIVES - a VAR integer *)

(* RETURNS - the ordinal position in a Working Record of the desired key *)

(* MAIN_MENU-4, RECS_MENU-3.1 Calls - N/A *)

**PROCEDURE LOAD_ANY_ARRAY (TotNames, WhichKey : integer);**

(* This procedure resets the Working Records File, reads it to the first *)

(* position of the sort key, and loads it and every 16th field following *)

(* it to the array which is otherwise used to hold names from NAMES.REC. *)

(* RECEIVES - two integers, one representing total names and the other *)

(* representing the record position of the field to sort on *)

(* RETURNS - an array of dates *)

(* MAIN_MENU-4, RECS_MENU-3.2 Calls - N/A *)

**PROCEDURE END_FOR_END (VAR TotNames : integer);**

(* for sorting. After the array is sorted, the procedure is called twice*)

(* more to return that date to its original condition, so that it may be *)

(* used as an index in sorting the Working Records File. *)

(* MAIN_MENU-4, RECS_MENU-3.3&3.5&3.6 Calls - N/A *)

**PROCEDURE SHELLSORT (TotNames : integer);**

(* This procedure is a Shellsort taken directly from page 131 of the book*)

(* DATA STRUCTURES OF COMPUTER INFORMATION SYSTEMS by Dr. Roy S. Ellzey. *)

(* It sorts in ascending order and uses Hibbard's Refinement. It is used *)

(* for sorting the 'Nam' array which contains various keys, depending on *)

(* the path by which this procedure is reached. *)

(* RECEIVES - an integer representing total names *)

(* RETURNS - a sorted array *)

(* MAIN-3 / MAIN_MENU-4, RECS_MENU-3.4 / MAIN_MENU-4, RECS_MENU-4.2 *)

(* Calls - EXCHANGE *)

**P-35**
PROCEDURE EXCHANGE (ExNdx, Distance : integer);
(* This procedure is used with SHELLSORT, performing the exchange and *)
(* backup. *)
(* RECEIVES — two integers, the exchange index and the distance (in *)
  record positions) to compare records *)
(* RETURNS — a more nearly sorted array *)
(* MAIN-3, SHELLSORT-1 )
(* MAIN_MENU-4, RECS_MENU-3.4, SHELLSORT-1 )
(* MAIN_MENU-4, RECS_MENU-4.2, SHELLSORT-1 Calls - N/A *)

*******************************************************************************
PROCEDURE GET_ANY_ORDER_INFO (TR : PRectype; TotNames, WhichKey : integer);
(* This procedure checks every record in the Working Records file against*)
(* the next value in a sorted array, formats the information in record *)
(* matches, and prints to the screen: Rname, Gendr, Bdate, Adate, Rdate, *)
(* HT, and PT. *)
(* RECEIVES — a working record and integers representing 1) total names *)
(* and 2) the position within the record of the sort field *)
(* MAIN_MENU-4, MENU_RECS-3.7 Calls - N/A *)

*******************************************************************************
PROCEDURE LOAD_NAME_ARRAY (TotNames : integer);
  AS SHOWN ABOVE
PROCEDURE LOAD_NAME_ARRAY (TotNames : integer);
  AS SHOWN ABOVE
PROCEDURE SHELLSORT (TotNames : integer);
  AS SHOWN ABOVE
PROCEDURE REWRITE_NAMEFILE (TotNames : integer);
  AS SHOWN ABOVE

*******************************************************************************
PROCEDURE GET_PERMA_ALPHA_INFO (TR : PRectype);
(* This procedure provides the same information from the Working Records *)
(* file as the previous option, only in this option it is presented in *)
(* alphabetical order by last names of those surveyed. A choice is given*)
(* the user of a printout or a screen display. As the letter drafting *)
(* features of the system can operate independently of the record keeping*)
(* functions, this procedure purposely uses the fields of the Names File *)
(* for the search keys. If they fail to exactly match the Rnames fields *)
(* from the Working Records File (which should never happen) it results *)
(* in a message to the user stating that the letter drafting features are*)
(* still operable, but records files are suspect and what notifications *)
(* should be made. *)

P-36
(* RECEIVES - a working record. *)
(* RETURNS - an alphabetized screen display or paper printout of Working *)
(* Records File information *)

(* MAIN_MENU-4, RECS_MENU-4.4 Calls - N/A *)

******************************************************************************
(* MAIN_MENU OPTION 4 Visit RECORD MANIPULATION MENU. *)
******************************************************************************
(* RECS_MENU OPTION 5 Seek help on this menu. *)
******************************************************************************

(* PROCEDURE ReadInst (x : integer); AS SHOWN ABOVE *)

******************************************************************************
(* MAIN_MENU OPTION 4 Visit RECORD MANIPULATION MENU. *)
******************************************************************************
(* RECS_MENU OPTION 6 Return to the MAIN MENU. *)
******************************************************************************
(* MAIN_MENU OPTION 5 Visit NORMING AND COMPARING MENU. *)
******************************************************************************

PROCEDURE NORM_MENU (VAR CR : PRectype; VAR Cal : Calctype);
(* This procedure contains all options for norming and/or comparing data.*)
(* MAIN_MENU-5 *)

VAR

    AnsGate: integer ;  (* For proper setting in PROCEDURE GetAnswers *)
    NorSet : char  ;  (* For registering choice of options *)
    Nkal  : Calctype ;  (* For collecting calculated values *)
    InName   ;  (* Required parameter for READ_WORKING_RECORD *)
    NFname  : string ;  (* For passing name of norming/normed file *)
    AA, BB ,  (* AA=Name, BB=Score Date, CC=DOB, DD=Test Date*)
    CC, DD : boolean ;  (* Indicate items desired to go in norming file*)

BEGIN
    REPEAT
        clrscr;
        WRITELN;
        WRITELN;
        WRITELN (' NORMING AND COMPARING MENU');
        WRITELN (' ENTER If You Wish To -');
        WRITELN;
        WRITELN (' 1 Create a standard, or norming, file.');
        WRITELN (' 2 Normalize an existing file.' );
        WRITELN (' 3 Display normalized data.');
        WRITELN (' 4 Compare normalized data with a record from file.' );
        WRITELN (' 5 Compare normalized data with current data.' );
        WRITELN (' 6 Add to a standard, or norming, file.' );
        WRITELN (' 7 Seek help on this menu.' );
        WRITELN (' 8 Return to the MAIN MENU.');
WRITELN;
WRITE ('   ');
READLN (NorSet);
IF NorSet = '1' THEN
BEGIN
  INDICATE_INFO_FOR_NORM_FILE (CR, AA, BB, CC, DD);
  NAME_NORM_FILE (Nfname);
  GetRSpsFile (NORMfile, Nfname);
  BUILD_NORM_FILE (CR, AA, BB, CC, DD);
END; (* IF *)
IF NorSet = '2' THEN
BEGIN
  INITIALIZE (Nkal);
  DO_THE_STATS (CR, Nkal, Nfname);
END; (* IF *)
IF NorSet = '3' THEN
BEGIN
  GET_ASSIGNED (Nfname);
  DISPLAY_NORMED_DATA (Nfname);
END; (* IF *)
IF NorSet = '4' THEN
BEGIN
  INITIALIZE (Cal);
  clrscr;
  AnsGate := 3;
  InName := 'NN'; (* To block calling CONVERT *)
  READ_WORKING_RECORD (CR, InName);
  IF CR.Rname <> 'ZZZZ' THEN
  BEGIN
    GET_Qans_ARRAY (CR);
    GetAnswers (CR, AnsGate, Cal);
    GET_ASSIGNED (Nfname);
    COMPARE_NORMED_DATA (Nfname, Cal, CR);
  END; (* IF *)
  END; (* IF *)
IF NorSet = '5' THEN
BEGIN
  clrscr;
  IF CR.Pname = ' ' THEN
  BEGIN
    WRITELN
    ( ' You have not loaded a record for comparison.' );
    WRITELN;
    WRITELN
    ( ' Recommend calling ''Read about PlugForm.''' );
    WRITELN;
    WRITELN
    ( ' Hit ENTER to return to menu.' );
    READLN;
  END (* IF *)
  END; (* IF * )
ELSE
    BEGIN
        GET_ASSIGNED (NFname);
        COMPARE_NORMED_DATA (NFname, Cal, CR);
        END; (* ELSE *)
    END; (* IF *)
IF NorSet = '6' THEN
    BEGIN
        clrscr;
        GET_ASSIGNED (NFname);
        RESET (NORMfile);
        READ_NORMING_FILE (CR);
        IF CR.Rname = 'New Record' THEN
            BEGIN
                NORM_APPEND_TELLS (CR, AA, BB, CC, DD);
                APPEND (NORMfile);
                BUILD_NORM_FILE (CR, AA, BB, CC, DD);
                END; (* IF *)
        ELSE
            BEGIN
                CLOSE (NORMfile);
                WRITELN;
                WRITELN ('The file you have chosen is not a ' +
                        'norming file, but you haven''t');
                WRITELN ('hurt anything. Hit enter and try again.');
                READLN;
                END; (* ELSE *)
            END; (* IF *)
        IF NorSet = '7' THEN ReadInst (11);
        UNTIL NorSet = '8';
    END; (* NORM_MENU *)

(*******************************************************************************)
(* NORM_MENU OPTION 1  Create a standard, or norming, file.  *)
(*******************************************************************************)
PROCEDURE INDICATE_INFO_FOR_NORM_FILE (VAR CR : Prectype;
                                          VAR AA, BB, CC, DD : boolean);
(* This procedure prompts user to set the parameters to be captured when *)
(* setting up a new Norming File. It provides options to capture Names, *)
(* Scoring Dates, Dates of Birth, and Testing Dates, or to leave any or *)
(* all of them out. *)
(* RECEIVES - a VAR Working Record and four VAR boolean variables *)
(* RETURNS - a Working Record with those optional fields the user skipped*)
(* set to 'N/A' and four boolean variables with values set *)
(* MAIN_MENU-5, NORM_MENU-1.1 Calls - N/A *)
PROCEDURE NAME_NORM_FILE (VAR NFname : string);
(* This procedure allows users to input a name for a new Norming File. *)
(* RECEIVES – a VAR string  *)
(* RETURNS – a string holding the name for a new norming file  *)
(* MAIN_MENU-15, NORM_MENU-1.2  Calls – N/A *)

PROCEDURE GetRspFile (VAR GOFile : text; VAR EnName : string);
  AS SHOWN ABOVE

PROCEDURE BUILD_NORM_FILE (VAR CR : Prectype; AA, BB, CC, DD : boolean);
(* This procedure calls the procedures, most requiring user input, for *)
(* those fields the user indicated a desire to capture, as well as the *)
(* required fields. "N/A" is written to field which are both optional and*)
(* unwanted. It repeats until the user indicates that it should stop.  *)
(* It is used for both the creation and the enlargement of norming files.*)

(* RECEIVES – a VAR Working Record and 4 booleans indicating fields to  *)
(*    be captured  *)
(* RETURNS – a full set of record fields written to the norming file  *)

(* MAIN_MENU-5, NORM_MENU-1.4  /  MAIN_MENU-5, NORM_MENU-6.4  *)

:* Calls – INITIALIZE, GetNames, GetToday, GetAnswers, MAKE_Qans_STRING, *
(*       GetGendr, NORM_BIRTHDAY, NORM_TEST_DATE        *)

PROCEDURE INITIALIZE (VAR Kal : Calctype);
  AS SHOWN ABOVE

PROCEDURE GetNames (VAR TR : Prectype; TotNames : integer;
  VAR Continuing : boolean; VAR EnName : string);
  AS SHOWN ABOVE

PROCEDURE GetChecks (TR : Prectype; TotNames : integer;
  VAR Continuing : boolean);
  AS SHOWN ABOVE

PROCEDURE GetToday (VAR TR : Prectype);
  AS SHOWN ABOVE

PROCEDURE GetAnswers (VAR TR : Prectype; AnsGate : integer;
  VAR Kal : Calctype);
  AS SHOWN ABOVE

PROCEDURE DISPLAY (JR : Prectype);
  AS SHOWN ABOVE
PROCEDURE GetAnsFast (VAR SR : PRectype; SS : integer);  

PROCEDURE CONVERT (b : integer);  

PROCEDURE TALLY (VAR C : Calctype);  

PROCEDURE GetGendr (VAR TR : PRectype);  

PROCEDURE NORM_BIRTHDAY (VAR TR : PRectype);  
(* This procedure prompts the user for the date of birth of the current *)  
(* entry into a norming file. It passes the user's response to ChekDate.*)  
(* If ChekDate does not accept it as valid, the reason is displayed in *)  
(* this procedure and the user may reenter the date. Otherwise, the date*)  
(* is set to the Working Record and returned. *)

(* RECEIVES - a VAR Working Record *)  
(* RETURNS - a Working Record with the value of Bdate set *)

(* MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-7 *)  
(* MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-7 *)  
(* Calls - ChekDate *)

PROCEDURE ChekDate (VAR date, why : string; VAR done : boolean);  

PROCEDURE NORM_TEST_DATE (VAR TR : PRectype);  
(* This procedure prompts the user for the testing date of the current *)  
(* entry into a norming file. It passes the user's response to ChekDate.*)  
(* If ChekDate does not accept it as valid, the reason is displayed in *)  
(* this procedure and the user may reenter the date. Otherwise, the date*)  
(* is set to the Working Record and returned. For purposes of setting up*)  
(* a norming file, the Rdate on the Working Record is used for test date.*)

(* RECEIVES - a VAR Working Record *)  
(* RETURNS - a Working Record with the value of Rdate set *)

(* MAIN_MENU-5, NORM_MENU-1.4, BUILD_NORM_FILE-8 *)  
(* MAIN_MENU-5, NORM_MENU-6.4, BUILD_NORM_FILE-8 *)  
(* Calls - ChekDate *)

PROCEDURE ChekDate (VAR date, why : string; VAR done : boolean);  

AS SHOWN ABOVE
(* MAIN_MENU OPTION 5  Visit NORMING AND COMPARING MENU. *)
(* NORM_MENU OPTION 2  Normalize an existing file. *)

PROCEDURE INITIALIZE (VAR Kal : Calctype);

PROCEDURE DO_THE_STATS (VAR CR : Prrectype; VAR Nkal : Calctype;
    VAR NFname : string);

(* This procedure calls all procedures required for taking an existing *)
(* file, reading and scoring each record in that file, determining the *)
(* mean, variance, and standard deviation for the entire file, allowing *)
(* the user to store that data to a new file, & displaying its contents. *)

(* RECEIVES - a Working Record, a Calculated Values Record, and a string *)
(* RETURNS - a normalized file and a screen display of that file *)

(* MAIN_MENU-5, NORM_MENU-2.2 *)
(* Calls - INITIALIZE, GET_ASSIGNED, READ_NORMING_FILE, GET_Qans_ARRAY, *)
(* TALLY, TOTE_STATS, GET_MEAN_ETAL, and DISPLAY_NORMED_DATA *)

VAR
    HdSums,  (* Holder for sums in norming *)
    HdSqrs : Calctype;  (* Holder for squares in norming *)
    count : integer;  (* For passing of which file name to norm *)

BEGIN
    INITIALIZE (HdSums);
    INITIALIZE (HdSqrs);
    count := 0;
    GET_ASSIGNED (NFname);
    RESET (NORMfile);
    REWRITE (NORMdone);
    WHILE NOT eof (NORMfile) DO
        BEGIN
            count := count + 1;
            READ_NORMING_FILE (CR);
            GET_Qans_ARRAY (CR);
            TALLY (Nkal);
            TOTE_STATS (Nkal, HdSums, HdSqrs);
            INITIALIZE (Nkal);
            END;  (* WHILE *)
            GET_MEAN_ETAL (HdSums, HdSqrs, count);
            CLOSE (NORMfile);
            RESET (NORMdone);
            DISPLAY_NORMED_DATA (NFname);
        END;  (* DO_THE_STATS *)
PROCEDURE INITIALIZE (VAR Kal : Calctype);

PROCEDURE GET_ASSIGNED (VAR NFname : string);
(*) This procedure allows the user to name a specific file or choose one *
(*) of four default files, Working Records, Male Norming, Female Norming, *
(*) or Combined Norming for normalization. If the user names a specific *
(*) file, error checking is disabled while the named file is RESET and, *
(*) if it does not exist, the user is so informed and given advice as to *
(*) how to proceed.

(*) RECEIVES - a VAR string  
(*) RETURNS - the name of the file in which the normed data will reside *

(*) MAIN_MENU-5, NORM_MENU-2.2, DO_THE_STATS-3 (INITIALIZE is 1 & 2) / *
(*) MAIN_MENU-5, NORM_MENU-3.1 / *
(*) MAIN_MENU-5, NORM_MENU-4.6 / *
(*) MAIN_MENU-5, NORM_MENU-5.1 / *
(*) MAIN_MENU-5, NORM_MENU-6.1 Calls - N/A *

PROCEDURE READ_NORMING_FILE (VAR CR : PRectype);
(*) Procedure reads one Working Record from the file being normed. *

(*) RECEIVES - a VAR Working Record  
(*) RETURNS - a Working Record with all fields filled *

(*) MAIN_MENU-5, NORM_MENU-2.2, DO_THE_STATS-4 / *
(*) MAIN_MENU-5, NORM_MENU-6.2 Calls - N/A *

PROCEDURE GET_Qans_ARRAY (TR : PRectype);

PROCEDURE TALLY (VAR C : Calctype);

PROCEDURE TOTE_STATS (VAR Nkal, HdsSums, HdsSqrS : Calctype);
(*) This procedure sums an individual person's survey evaluation (and its *)
(*) square) into running totals.

(*) RECEIVES - 3 VAR Calculated Values Records, 1 holding the calculate *
(*) values for the current surveyed person, 1 the running total*)
(*) of the sum of its numeric values, and 1 holding the running*)
(*) totals of the squares of those values *
(*) RETURNS - updated values in the later two of the above *

(*) MAIN_MENU-5, MN-2.2, DO_THE_STATS-7 Calls - N/A *
PROCEDURE INITIALIZE (VAR Kal : Calctype);

PROCEDURE GET_MEAN_Etal (VAR HdSums, HdSqrss: Calctype; VAR count: integer);
(* This procedure calculates Mean, Variance, and Standard Deviation and, *)
(* in turn, writes the lead field of each to the normed file and calls *)
(* WRITE_NORMED_DATA to store the eleven relevant values calculated. *)

(* RECEIVES – two Calculated Values Records, one with the sum of the all *)
(* numeric values and one with the sum of their squares, and *)
(* an integer holding a count of records processed *)
(* RETURNS – a normed file of the same name as the norming file, but a *)
(* suffix of nom *)

(* MAIN_MENU-5, MN-2.2, DO_THE_STATS-9 *)
(* Calls – WRITE_NORMED_DATA *)

VAR
   MEAN , /* Record of relevant mean values in normed file. */
   VARY , /* Record of variances from mean in normed file. */
   STDV : Calctype; /* Record of standard deviations in normed file. */

BEGIN
   REWRITE (NORMdone);
   INITIALIZE (MEAN);
   INITIALIZE (VARY);
   INITIALIZE (STDV);
   WITH MEAN DO
      BEGIN
         D := HdSums.D / count;
         DP := HdSums.DP / count;
         D1 := HdSums.D1 / count;
         A := HdSums.A / count;
         AP := HdSums.AP / count;
         P := HdSums.P / count;
         PP := HdSums.PP / count;
         P1 := HdSums.P1 / count;
         EX1 := HdSums.EX1 / count;
         EXP := HdSums.EXP / count;
         END; (* WITH *)
   WRITELN (NORMdone, 'MEAN');
   WRITE_NORMED_DATA (MEAN);
WITH VARY DO
BEGIN
    D := (HdSgrs.D - (((SQR(HdSums.D))/count)))/(count - 1);
    DP := (HdSgrs.DP - (((SQR(HdSums.DP))/count)))/(count - 1);
    D1 := (HdSgrs.D1 - (((SQR(HdSums.D1))/count)))/(count - 1);
    A := (HdSgrs.A - (((SQR(HdSums.A))/count)))/(count - 1);
    AP := (HdSgrs.AP - (((SQR(HdSums.AP))/count)))/(count - 1);
    A1 := (HdSgrs.A1 - (((SQR(HdSums.A1))/count)))/(count - 1);
    P := (HdSgrs.P - (((SQR(HdSums.P))/count)))/(count - 1);
    PP := (HdSgrs.PP - (((SQR(HdSums.PP))/count)))/(count - 1);
    P1 := (HdSgrs.P1 - (((SQR(HdSums.P1))/count)))/(count - 1);
    EX1 := (HdSgrs.EX1 - (((SQR(HdSums.EX1))/count)))/(count - 1);
    EXP := (HdSgrs.EXP - (((SQR(HdSums.EXP))/count)))/(count - 1);
END; (* WITH *)
WRITELN (NORMdone, 'VARIANCE');
WRITE_NORMED_DATA (VARY);
WITH STDV DO
BEGIN
    D := SQR(VARY.D);
    DP := SQR(VARY.DP);
    D1 := SQR(VARY.D1);
    A := SQR(VARY.A);
    AP := SQR(VARY.AP);
    A1 := SQR(VARY.A1);
    P := SQR(VARY.P);
    PP := SQR(VARY.PP);
    P1 := SQR(VARY.P1);
    EX1 := SQR(VARY.EX1);
    EXP := SQR(VARY.EXP);
END; (* WITH *)
WRITELN (NORMdone, 'DEVIATION');
WRITE_NORMED_DATA (STDV);
WRITELN (NORMdone, 'This norming file based on ', count, ' records.');
CLOSE (NORMdone);
END; (* GET_MEAN_ETAL *)

(******************************************************************
PROCEDURE WRITE_NORMED_DATA (KALK : Calctype);
(* This procedure writes normed data to a previously named normed data *)
(* file. *)

(* RECEIVES - a Calculated Values Record)
(* RETURNS - an updated, finally completed, normed data file *)

(* MAIN_MENU-5, NORM_MENU-2.2, DO_THE_STATS-9, GET_MEAN_ETAL-4&5&6 *)
(* Calls - N/A *)

P-45
PROCEDURE DISPLAY_NORMED_DATA (NFname : string);
(* This procedure calls READ_NORMED_DATA for Calculated Values Records *)
(* from a previously specified normed data file, formats the values, and *)
(* displays that data to the screen. *)
(* RECEIVES – a string containing user designated name of the normed file *)
(* to be displayed *)
(* RETURNS – a formatted screen display of values from the required file *)

(* MAIN_MENU-5, NORM_MENU-2.2, DO_THE_STATS-10 / *)
(* MAIN_MENU-5, NORM_MENU-3.2 / *)
(* MAIN_MENU-5, NORM_MENU-5.2 *
(* Calls – READ_NORMED_DATA *)

PROCEDURE READ_NORMED_DATA (VAR CALC : Calctype);
(* Procedure calls PROCEDURE INITIALIZE for Calctype record and reads the *)
(* record at the data pointer from the normed file. *)
(* RECEIVES – a VAR Calculated Values Record *)
(* RETURNS – a Calculated Values Record with required values filled *)

(* MAIN_MENU-5, MN-2.2, DO_THE_STATS-10, DISPLAY_NORMED_DATA-1&2&3 / *)
(* MAIN_MENU-5, NORM_MENU-3.2, DISPLAY_NORMED_DATA-1&2&3 / *)
(* MAIN_MENU-5, NORM_MENU-4.7, COMPARE_NORMED_DATA-1&2&3 / *)
(* MAIN_MENU-5, NORM_MENU-5.2, COMPARE_NORMED_DATA-1&2&3 *
(* Calls – INITIALIZE *

PROCEDURE INITIALIZE (VAR Kal : Calctype);

************ AS SHOWN ABOVE

******************************************************************************
(* MAIN_MENU OPTION 5 Visit NORMING AND COMPARING MENU. *)
******************************************************************************
(* NORM_MENU OPTION 3 Display normalized data. *)
******************************************************************************

PROCEDURE GET_ASSIGNED (VAR NFname : string);

************ AS SHOWN ABOVE

******************************************************************************
PROCEDURE DISPLAY_NORMED_DATA (NFname : string);

************ AS SHOWN ABOVE

******************************************************************************
PROCEDURE READ_NORMED_DATA (VAR CALC : Calctype);

************ AS SHOWN ABOVE

******************************************************************************
PROCEDURE INITIALIZE (VAR Kal : Calctype);

************ AS SHOWN ABOVE

P-46
(* MAIN_MENU OPTION 5  Visit NORMING AND COMPARING MENU. *)

(* NORM_MENU OPTION 4  Compare normalized data with a record from file. *)

PROCEDURE READ_WORKING_RECORD (VAR TR : PRectype; VAR InCome : string);
    AS SHOWN ABOVE

PROCEDURE CoverDups (VAR TR : PRectype);
    AS SHOWN ABOVE

PROCEDURE GET_Qans_ARRAY (TR : PRectype);
    AS SHOWN ABOVE

PROCEDURE GetAnswers (VAR TR : PRectype; AnsGate : integer;
    VAR Kal : Calctype);
    AS SHOWN ABOVE

PROCEDURE DISPLAY (JR : PRectype);
    AS SHOWN ABOVE

PROCEDURE GetAnsSlow (VAR SR : PRectype);
    AS SHOWN ABOVE

PROCEDURE GetAnsFast (VAR SR : PRectype; SS : integer);
    AS SHOWN ABOVE

PROCEDURE CONVERT (b : integer);
    AS SHOWN ABOVE

PROCEDURE TALLY (VAR C : Calctype);
    AS SHOWN ABOVE

PROCEDURE GET_ASSIGNED (VAR NName : string);
    AS SHOWN ABOVE

PROCEDURE COMPARE_NORMED_DATA (VAR NName : string; VAR Cal : Calctype;
    VAR PR : PRectype);

(* This procedure compares values calculated from a Working Record to *)
(* normed data from a Normed Data file specified by the user, displaying *)
(* the comparison to the screen in a formatted manner. *)

(* RECEIVES - the current Working Record, the current Calculated Values *)
(* Record, and a string containing the name of the normed *)
(* data file being used for comparison *)
(* RETURNS - a screen output of the scores and T scores of the person *)
(* specified, together with the Mean, Variance, and Standard *)
(* Deviation of those scores from the normed file specified *)
(* MAIN_MENU-5, NORM_MENU-4.7 *)
(* Calls - READ_NORMED_DATA *)
PROCEDURE READ_NORMED_DATA (VAR CALC : Calctype);

(* MAIN_MENU OPTION 5  Visit NORMING AND COMPARING MENU. *)
(* NORM_MENU OPTION 6  Add to a standard, or norming, file. *)

PROCEDURE GET_ASSIGNED (VAR Nfilename : string);

PROCEDURE NORM_APPEND_TELLS (CR : Prectype; VAR AA, BB, CC, DD : boolean);
(* This procedure is used to set the parameters to be captured when one *)
(* desires to add to an existing Norming File. It looks at the first *)
(* record to determine which fields were captured in the existing file *)
(* & sets the boolean values to capture the same fields while appending. *)

(* RECEIVES - a Working Record set to the values of the first record in *)
(* the norming file being appended and four VAR boolean variables*)
(* RETURNS - 4 boolean variables set regarding fields to be captured *)

(* MAIN_MENU-5, NORM_MENU-6.3 *)

PROCEDURE BUILD_NORM_FILE (VAR CR : Prectype; AA, BB, CC, DD : boolean);

PROCEDURE INITIALIZE (VAR Kal : Calctype);

PROCEDURE GetNames (VAR TR : Prectype; TotNames : integer;
  VAR Continuing : boolean; VAR EnName : string);

PROCEDURE GetChecks (TR : Prectype; TotNames : integer;
  VAR Continuing : boolean);

PROCEDURE GetToday(VAR TR :Prectype);

PROCEDURE GetAnswers (VAR TR : Prectype; AnsGate : integer;
  VAR Kal : Calctype);

PROCEDURE DISPLAY (JR : Prectype);

PROCEDURE GetAnsFast (VAR SR : Prectype; SS : integer);

P-48
PROCEDURE DISPLAY (JR : PRectype);

PROCEDURE CONVERT (b : integer);

PROCEDURE INITIALIZE (VAR Kal : Calctype);

PROCEDURE TALLY (VAR C : Calctype);

PROCEDURE GetGendr (VAR TR : PRectype);

PROCEDURE NORM_BIRTHDAY (VAR TR : PRectype);

PROCEDURE CheckDate (VAR date, why : string; VAR done : boolean);

PROCEDURE NORM_TEST_DATE (VAR TR : PRectype);

PROCEDURE CheckDate (VAR date, why : string; VAR done : boolean);

(* MAIN_MENU OPTION 5   Visit NORMING AND COMPARING MENU. *)
(* NORM_MENU OPTION 7    Seek help on this menu. *)
(* PROCEDURE ReadInst (x : integer); *)

(* MAIN_MENU OPTION 5   Visit NORMING AND COMPARING MENU. *)
(* NORM_MENU OPTION 8   Return to the MAIN MENU. *)
(* MAIN_MENU OPTION 6   Quit PLUGFORM program. *)
WordPerfect MACROS:

There are 18 WordPerfect macros supporting the PLUGFORM system. Six of these macros are visible to the user and the remaining 12 are chained to one of the others (macro ALT V.) Brief descriptions of these macros follow:

The ALT U macro is executed by the user upon shifting to WordPerfect. It calls to the screen a listing of all floppy disk files with "??L" suffixes, which are only response letter files.

After retrieving the letter of their choice, users enter ALT V. The ALT V macro searches the response letter file for the trigger letter sequence "AAAA" and, if "AAAA" is found, removes that letter sequence, searches for the heading of the second page, inserts the page break, goes to the bottom of the page, adds a specific paragraph which is governed by the matrix in program PROCEDURE BIGIF (where the trigger letter sequence was added to the response letter) and chains into macro LASTPASS.

LASTPASS adds a last paragraph, which is the same for all response letters, and the closing and the signature block, and chains to macro HEREDOC.

HEREDOC searches for the trigger letter sequence "DOTHEDOC" and, if found, searches for both "attorney" and "client" and replaces them with "physician" and "patient". HEREDOC chains to SPHELL, if "DOTHEDOC" is found and back chains to SPHELL, if it is not. SPHELL puts the user into the WordPerfect Spelling Check.

If letter sequence "AAAA" was not found by the search initiated by ALT V, the macro back chains to macro BBBB which searches for the letter sequence "BBBB". A successful search initiates the same sequence of events described for finding letter sequence "AAAA", excepting that a different paragraph is generated. Search failure causes a back chain into macro DDDD.

Macro DDDD acts in a similar manner to BBBB and back chains to Macro EEEE, which is also similar and back chains to macro FFFF. Macro FFFF works in the same way and back chains to macro HHHH. Thus are six (of the eight possible results from the matrix in PROCEDURE BIGIF) of the situations supported. Macro HHHH back chains into macro CCCC and, if letter sequence "CCCC" is found, a message is printed to the screen listing two different paragraphs, with the respective single key macros (ALT A and ALT C) which will generate those paragraphs and chain to LASTPASS. Macro CCCC and macro GGGG both chain to macro XXXX, which displays the caution that the choice of the displayed paragraphs is to be made (and initiated for) by a PHD. Macro CCCC back chains into macro GGGG which, in a similar fashion, displays two more paragraphs (generated by ALT E and ALT G, respectively) as well as chaining to macro XXXX.

Failure to conclude a successful search, which should never happen, will back chain into macro AAAA. Macro AAAA generates a message stating that the program is not operating properly and outlines the proper course of actions to initiate.

P-50
RESULTS AND CONCLUSIONS

The PLUGFORM system can evaluate survey results, draft or reconstruct (from the Working Records File) both letters and work sheets, create records and automatically file them, sort and display records, correct records, delete records, create norming files, append norming files, normalize files, display normed data, display the comparisons of an individual's test scores against normalized data (including the calculation of individual "Z scores") and provide the data for norming the survey and a basis for judging when a survey should be renormed.

The PLUGFORM system produces letters and work sheets conforming to the eight basic situations outlined by the company and listed in the addendum to the introduction. Examples of these may be seen in Appendix B, pages 1 through 24. Further examples of PLUGFORM products include:


2) Normed files, Appendix B, pages 29 through 32.

3) Particular survey results compared to particular normed files, Appendix B, pages 33 through 36.

The letter and work sheet formats do not lend themselves to subjective judgement, as examples were provided as to what they should look like and the letters and work sheets which PLUGFORM produces conform to those examples. Speed is of not much concern in the production of the letters and work sheets. After an initial delay of about six seconds in booting, there is little to no "dead time" noticeable to the user while creating them. There are several stoppages in the creation of a letter, where the user is queried as to a desire to start the section or, in places, the whole process over. This is a trade off between not punishing the careless user, who starts a process in error, by forcing a section's continuance to its natural end, and not driving the highly competent user to distraction by forcing a "yes we are okay" input after every step. The company has had the letter writing section of PLUGFORM for several months and states that they are pleased with it. This section is evaluated as having been a highly successful effort.

The results of the record keeping section are more mixed. There is no problem with the filing and retrieval of records; the filing is invisible and the retrieval time is less than 2 seconds. Tests were run on record retrieval and all RECORDS MANIPULATION MENU sections using the below listed procedure:
PROCEDURE Time_It (VAR Time : string);
(* This procedure captures the time from the cpu registers. *)

VAR
REGS : registers; (* accesses cpu registers *)
HH, MM, SS : string [2]; (* hours, minutes, seconds *)

BEGIN
REGS.AX := $2C00;
MSDOS(REGS);
STR(REGS.CH:2,HH);
STR(REGS.CL:2,MM);
STR(REGS.DH:2,SS);
TIME := HH + ':' + MM + ':' + SS;
END; (* Time_It *)

Tests were run using a Working Records File of 155 records.

Tests run using a 486 PC curtailed the CAddr, ADisc, and AdSymp fields by 80%, with no discernable differences in the run times.

The record correction and the record deletion options took an average of 38 seconds and 40 seconds, respectively, when run from the floppy drive of a 486 PC. Tests run from the floppy drive of a 386 PC averaged a second longer for each. Tests run from the floppy drive of a 286 machine took about twice as long, but the testing was not extensive enough to obtain an accurate average.

Sorts (Options 3 and 4 of the RECORDS MANIPULATION MENU) were timed using sorts to the screen and "commenting out" the break between each 20 lines of display. The 486 PC, running from the floppy drive, could sort and display in an average of 8 seconds, the 386 PC in 20 seconds, and the 286 PC in 7 to 8 minutes. This last figure was considered to be disappointing, however:

1) The company stated a maximum figure of 150 records in the Working Records File, and indicated they thought a figure that high to be optimistic.

2) The company seemed not much interested in an ability to sort records. (Realistically, they must have a capability to look at the names within their Working Records File, but this could easily be obtained by displaying the index loaded from NAMES.REC to the screen.)

3) A viable option would be to run all records on the hard drive. (This was tried, and cut sort time to about 25 seconds.)

4) If all else failed, the use of text files could be abandoned.

5) This section has yet to be observed while operating on the company's PC. (The one 286 it has been observed on is now inoperable.)

R-2
The options reached from the NORMING AND COMPARING MENU are more important to the needs of the company than those of the previous section. All of these options ran well on every PC on which they were tested. The most time consuming option on the NORMING AND COMPARING MENU is to normalize an existing file. For normalization, times ran from an average of .5 seconds for the 486 PC to normalize an 88 record file to a longest time expended of 9 seconds for the 286 PC to normalize a 220 record file. Times consumed in executing other options on this menu ranged between less than one second and three seconds.

Based partly upon the results of normalizing several batches of tests given for standardization, the company altered the survey. Several batches of the new surveys, totalling 220, were normalized with PLUGFORM and resulted in not only providing the cutoff setting for the proposed test, but also in changing the methodology of comparison. At the company's request, the system is now set to formally score individuals against only same sex norming samples. The NORMING AND COMPARING MENU allows a psychologist to bring several comparisons to the screen (which the company indicates would be done in less clear cut cases) but the response letter and work sheet reflect the more formal score. PLUGFORM is not capable of having made any of these decisions for change, but it demonstrated the need for them as clearly as, and considerably faster than, the experts who provided its governing formulae could have done.

The ability for the user to easily create small, stand-alone, norming files is also of considerable value to the company. By taking periodic, smaller (cheaper) samples, for comparison with results from the internal, static, system cutoff points, early warnings may be provided of possibly evolving social attitudes. Maximum flexibility is provided by allowing the option of keeping such samples separate and/or appending them to existing files. Data may be studied at leisure and rejected or encompassed as desired. (I.e., it would be facile to have dynamic cutoff points, which changed with each addition to the norming sample, but they would be at risk to corruption by a temporary aberration or by a poorly administered test.)

An ultimate evaluation of PLUGFORM will lie in its interface with clerical users. The completed PLUGFORM system was tested using three individuals and, without extra-system instruction, all could use PLUGFORM within 30 minutes. While that is not an extensive sample, it does provide an indication.

A letter which indicates PSYCHOLOGICAL RESOURCES OF DALLAS's satisfaction with the system is contained at Appendix C.