

# St. Brendan's College Interim Reporting System

April 1996



Developed by  
Peter Doyle  
Simon Wagner  
Lucas Wyte

# ST BRENDAN'S COLLEGE

## INTERIM REPORTING SYSTEM

### PROGRESS LOG

|          |   |
|----------|---|
| 7 March  | Peter starts hand-drawn draft of Create procedure.<br>Simon starts hand-drawn drafts of SortName and SortPcnt procedures.<br>Lucas starts hand-drawn draft of main menu.  |
| 8 March  | Peter completed Create; starts hand-drawn draft of Load procedure.<br>Simon completed Sort's; starts hand-drawn draft of OutSummy procedure.<br>Lucas completed menu; starts hand-drawn draft of OutSlctd and   |
| OutAll   | procedures.   |
| 11 March | Peter completed Load; makes some changes to outputs in Create procedure.<br>Simon completes OutSummy procedure.<br>Lucas completed OutAll; completes OutSlctd with assistance of  |
| Simon.   |   |
| 12 March | Peter starts Delta draft of Load procedure.<br>Simon starts Delta drafts of SortName and SortPcnt procedures.<br>Lucas starts Delta draft of main menu and entry of global variables.   |
| 14 March | Peter completed Load; starts Delta draft of Create procedure.<br>Simon continuing drafts of SortName and SortPcnt procedures.<br>Lucas completed menu; starts Delta drafts of OutSlctd procedure.   |
| 18 March | Peter, on consultation with group, makes changes to Create procedure - changing role of procedure to that of an Append procedure.<br>Simon completed Sort's; starts Delta draft of OutSummy procedure.<br>Lucas continuing Delta draft of OutSlctd procedure. |
| 19 March | Peter edits Load procedure in conforming with new Append procedure.<br>Simon continuing Delta draft of OutSummy procedure.<br>Lucas completed OutSlctd; starts Delta draft of OutAll procedure.   |
| 20 March | Peter edits Load procedure to load as part of the main program rather than as a user-selectable procedure.<br>Simon continuing OutSummy procedure.<br>Lucas completes OutAll procedure.   |
| 21 March | Peter starts Delta drafts of TchFile and SubjFile procedures.<br>Simon completes OutSummy procedure.<br>Lucas enters test data into database; edits Out' procedures to suit data.   |

|          |   |
|----------|---|
| 25 March | Peter adds TchFile and SubjFile procedures to main menu; begins testing of outputs with new procedures producing favourable results. Simon begins testing of OutSummy procedure. Lucas begins initial coding and testing of database.   |
| 27 March | Peter continues testing of procedures and making necessary changes. Simon continues testing of procedures, concentrating on user-proofing. Lucas continues testing of procedures, making necessary changes  |
| while    | concentrating on user-friendliness and layout.  |
| 28 March | Most procedures operational; minor bugs being discussed by group in order to reach a solution.  |
| 16 April | Simon returns from holidays and discovers solution to OutSummy spacing bug; edits procedure accordingly. Peter returns from holidays and discovers solution to variable problem in Append and Load procedures; edits procedures accordingly. Lucas returns from holidays and discovers solution to text file storage problem; edits necessary procedures accordingly. |
| 17 April | Simon puts finishing touches on SortName and SortPcnt procedures; begins final testing. Peter puts finishing touches on Append and Load procedures; begins final testing. Lucas puts finishing touches on Out' procedures; begins final testing   |
| and      | coding.   |
| 19 April | System fully operational, coded and ready for printing of documentation.  |



## ST. BRENDAN'S COLLEGE

MARY'S MOUNT, YEPPOON. 4703.  
YEPPOON Q 4703

Student's Name: Liam DAY

Semester: TWO, 1995

Year Level: 11

Subject: I. P. T.

Teacher's Name: Mr. HAMILTON

### OVERALL LEVEL OF ACHIEVEMENT

#### CRITERIA

KNOWLEDGE & SIMPLE FAMILIAR

APPLICATIONS.....

SYNTHESIS, COMMUNICATION

ANALYSIS & EVALUATION.....

|                | Good | Satisfactory | Needs Improvement |
|----------------|------|--------------|-------------------|
| Attitude       |      |              |                   |
| Conduct        |      |              |                   |
| Work Presented |      |              |                   |
| Progress       |      |              |                   |

Comment: .....

.....  
.....  
.....  
.....  
.....

Teacher: ..... Pastoral Carer Mr. CONN

This program is designed as an interim reporting system which will advise parents of students performance in a particular subject. The reports will be printed on preprinted stationery. This menu was designed by Lucas

Begin the menu program, using pre-tested iteration.

Loop Initialisation

while SELECTION <> '7'

Output the menu, prompt the user to make a selection and run the corresponding procedure.

Assign  
'0' to S  
ELECTION

Clear  
the  
screen.

Load the  
school  
files.

Output  
the  
menu.

User-pro  
of menu,  
assignin  
g readkey  
to SELEC  
TION and

Select  
the corr  
espondin  
g proced  
ure from  
the  
input

Cl  
ea  
rS  
cr  
ee  
n(  
)

Load  
the  
stud  
ent  
file

Load  
the  
subj  
ect  
file

Load  
the  
teac  
her  
file

Ass  
ign  
rea  
dke  
y

case

unti  
l SE

O  
u  
t  
p  
u  
t

O  
u  
t  
p  
u  
t

O  
u  
t  
p  
u  
t

A  
s  
s  
i  
g  
n  
r

Identifiers used in program REPORTER (User Brendans 1 May 1996)

Type declarations (2)

Local STR25 = string[25];

Local STUDENT = record

SURNAME:STR25

NAME:STR25

YEAR:integer single

SUBJCODE:integer single

TEACHCODE:integer single

ACHIEVE:character

PERCENT:integer single

ATTITUDE:character

CONDUCT:character

PROGRESS:character

end; record

Variable declarations (6)

Local SELECTION : character;

Local COUNTER : integer single;

Local STUDFILE : array[1..30] of STUDENT;

Local STUDTEXT : textfile;

This procedure is designed to load the student file saved as STUDENTS.RPT into memory. This procedure was designed by Peter Doyle and Simon Wagner, and edited by Lucas Wyte for St. Brendan's College.

Load the student file.

Assign the text file to the file saved on disc.

Reset the file.

Read the file.

Close the file.

Assign(STUDENTTEXT, 'a:\students.rpt')

Reset(STUDENTTEXT)

Loop Initialisation

while NOT EOF(STUDENTTEXT)  
Read each record.

Close(STUDENTTEXT)

Assign 0 to COUNTER.

Adjust the counter.

Read the record.

Assign COUNTER + 1 to COUNTER.

Input STUDFILE[COUNTER].SURNAME,  
[newline], STUDFILE[COUNTER].NAME,  
[newline], STUDFILE[COUNTER].YEAR,  
[newline], STUDFILE

Input STUDFILE[COUNTER].ACHIEVE,  
[newline], STUDFILE[COUNTER].ATTITUDE,  
[newline], STUDFILE[COUNTER].CONDUCT,  
[newline], STUDFILE

Identifiers used in procedure LOADSTUD (User Brendans 1 May 1996)

Variable declarations (3)

Global STUDTEXT : textfile;

Global STUDFILE : array[1..30] of STUDENT;

Global COUNTER : integer single;

This procedure is designed to load the subject file saved as SUBJECTS.DAT into memory.

Load the subject file.

Assign the text file to the file saved on disc.

Reset the file.

Read the file.

Close the file.

Assign(  
SUBJTEXT, 'a  
:\subjects.  
dat')

Reset(  
SUBJTEXT)

For ELEMENT  
:= 1 to 20,  
step 1  
Read each  
subject.

Close(  
SUBJTEXT)

Input SUBJECT[ELEMENT], [newline] from  
SUBJTEXT.

Identifiers used in procedure LOADSUBJ (User Brendans 1 May 1996)

Variable declarations (6)

```
Global STUDTEXT : textfile;  
Global STUDFILE : array[1..30] of STUDENT;  
Global COUNTER : integer single;  
Local SUBJTEXT : textfile;  
Local ELEMENT : integer single;  
Global SUBJECT : array[1..20] of STR25;
```

This procedure is designed to load the teacher file saved as TEACHERS.DAT into memory.

Load the teacher file.

Assign the text file to the file saved on disc.

Reset the file.

Read the file.

Close the file.

Assign(TCHTEXT, 'a:\teachers.dat')

Reset(TCHTEXT)

For ELEMENT := 1 to 20, step 1

Read each teacher's name.

Close(TCHTEXT)

Input TEACHER[ELEMENT], [newline] from TCHTEXT.

Identifiers used in procedure LOADTCH (User Brendans 1 May 1996)

Variable declarations (8)

```
Global STUDTEXT : textfile;  
Global STUDFILE : array[1..30] of STUDENT;  
Global COUNTER : integer single;  
Local SUBJTEXT : textfile;  
Local ELEMENT : integer single;  
Global SUBJECT : array[1..20] of STR25;  
Local TCHTEXT : textfile;  
Global TEACHER : array[1..20] of STR25;
```



```

graph TD
    Start([Start]) --> ClearScren[ClearScren()]
    ClearScren --> PromptSec[Prompt "SECURITY CHECK -"]
    PromptSec --> VerifyP[Verify password and if the]
    VerifyP --> AssignNil[Assign "NIL" to PASSWORD.]
    AssignNil --> InputNew[Input [newline] from]
    InputNew --> IfPass{if PASSWORD = 'jesusluvsu'}
    IfPass --> PromptUser[then Prompt user to input details and]
    PromptUser --> OutputInc[Output '** PASSWORD INCORRECT **',]
    OutputInc --> AppendFile[Append the student file, saved on disk as STUDENTS.RPT.]
    AppendFile --> OutputNewLine[Output [newline], [newline], "Student file appended.",]
    OutputNewLine --> PerformPost[Perform a post-tested iteration, allowing]
    PerformPost --> PromptAgain[Prompt until AGAIN in ['N'],]
    PromptAgain --> AssignStud[Assign( STUDTEXT, 'a:\stude]
    AssignStud --> Rewrite[Rewrite( STUDTEXT)]
    Rewrite --> WriteArray[Write the contents of the array]
    WriteArray --> CloseStud[Close( STUDTEXT)]
    CloseStud --> ClearScreen[ClearScreen()]
    ClearScreen --> PromptInd[Prompt user to input individual students' details.]
    PromptInd --> ForLoop[For ELEMENT := 1 to COUNTER, step 1]
    ForLoop --> AppendStudent[Append the student]
    AppendStudent --> AssignCounter[Assign COUNTER + 1 to]
    AssignCounter --> PromptUser1[Prompt user to input]
    PromptUser1 --> PromptUser2[Prompt user to input]
    PromptUser2 --> PromptWish[Prompt "Do you wish to]
    PromptWish --> OutputS1[Output S TUDFILE[ ELEMENT]]
    OutputS1 --> OutputS2[Output S TUDFILE[ ELEMENT]]
    OutputS2 --> PromptPleas1[Prompt "Pleas e]
    PromptPleas1 --> PromptPleas2[Prompt "Pleas e]
    PromptPleas2 --> PromptPleas3[Prompt "Pleas e]
    PromptPleas3 --> PromptPleas4[Prompt "Pleas e]
    PromptPleas4 --> PromptPleas5[Prompt "Pleas e]
    PromptPleas5 --> PromptPleas6[Prompt "Pleas e]
    PromptPleas6 --> PromptPleas7[Prompt "Pleas e]
    PromptPleas7 --> PromptPleas8[Prompt "Pleas e]
    PromptPleas8 --> PromptPleas9[Prompt "Pleas e]
    PromptPleas9 --> PromptPleas10[Prompt "Pleas e]
    PromptPleas10 --> PromptPleas11[Prompt "Pleas e]
    PromptPleas11 --> PromptPleas12[Prompt "Pleas e]
    PromptPleas12 --> PromptPleas13[Prompt "Pleas e]
    PromptPleas13 --> PromptPleas14[Prompt "Pleas e]
    PromptPleas14 --> PromptPleas15[Prompt "Pleas e]
    PromptPleas15 --> PromptPleas16[Prompt "Pleas e]
    PromptPleas16 --> PromptPleas17[Prompt "Pleas e]
    PromptPleas17 --> PromptPleas18[Prompt "Pleas e]
    PromptPleas18 --> PromptPleas19[Prompt "Pleas e]
    PromptPleas19 --> PromptPleas20[Prompt "Pleas e]
    PromptPleas20 --> PromptPleas21[Prompt "Pleas e]
    PromptPleas21 --> PromptPleas22[Prompt "Pleas e]
    PromptPleas22 --> PromptPleas23[Prompt "Pleas e]
    PromptPleas23 --> PromptPleas24[Prompt "Pleas e]
    PromptPleas24 --> PromptPleas25[Prompt "Pleas e]
    PromptPleas25 --> PromptPleas26[Prompt "Pleas e]
    PromptPleas26 --> PromptPleas27[Prompt "Pleas e]
    PromptPleas27 --> PromptPleas28[Prompt "Pleas e]
    PromptPleas28 --> PromptPleas29[Prompt "Pleas e]
    PromptPleas29 --> PromptPleas30[Prompt "Pleas e]
    PromptPleas30 --> PromptPleas31[Prompt "Pleas e]
    PromptPleas31 --> PromptPleas32[Prompt "Pleas e]
    PromptPleas32 --> PromptPleas33[Prompt "Pleas e]
    PromptPleas33 --> PromptPleas34[Prompt "Pleas e]
    PromptPleas34 --> PromptPleas35[Prompt "Pleas e]
    PromptPleas35 --> PromptPleas36[Prompt "Pleas e]
    PromptPleas36 --> PromptPleas37[Prompt "Pleas e]
    PromptPleas37 --> PromptPleas38[Prompt "Pleas e]
    PromptPleas38 --> PromptPleas39[Prompt "Pleas e]
    PromptPleas39 --> PromptPleas40[Prompt "Pleas e]
    PromptPleas40 --> PromptPleas41[Prompt "Pleas e]
    PromptPleas41 --> PromptPleas42[Prompt "Pleas e]
    PromptPleas42 --> PromptPleas43[Prompt "Pleas e]
    PromptPleas43 --> PromptPleas44[Prompt "Pleas e]
    PromptPleas44 --> PromptPleas45[Prompt "Pleas e]
    PromptPleas45 --> PromptPleas46[Prompt "Pleas e]
    PromptPleas46 --> PromptPleas47[Prompt "Pleas e]
    PromptPleas47 --> PromptPleas48[Prompt "Pleas e]
    PromptPleas48 --> PromptPleas49[Prompt "Pleas e]
    PromptPleas49 --> PromptPleas50[Prompt "Pleas e]
    PromptPleas50 --> PromptPleas51[Prompt "Pleas e]
    PromptPleas51 --> PromptPleas52[Prompt "Pleas e]
    PromptPleas52 --> PromptPleas53[Prompt "Pleas e]
    PromptPleas53 --> PromptPleas54[Prompt "Pleas e]
    PromptPleas54 --> PromptPleas55[Prompt "Pleas e]
    PromptPleas55 --> PromptPleas56[Prompt "Pleas e]
    PromptPleas56 --> PromptPleas57[Prompt "Pleas e]
    PromptPleas57 --> PromptPleas58[Prompt "Pleas e]
    PromptPleas58 --> PromptPleas59[Prompt "Pleas e]
    PromptPleas59 --> PromptPleas60[Prompt "Pleas e]
    PromptPleas60 --> PromptPleas61[Prompt "Pleas e]
    PromptPleas61 --> PromptPleas62[Prompt "Pleas e]
    PromptPleas62 --> PromptPleas63[Prompt "Pleas e]
    PromptPleas63 --> PromptPleas64[Prompt "Pleas e]
    PromptPleas64 --> PromptPleas65[Prompt "Pleas e]
    PromptPleas65 --> PromptPleas66[Prompt "Pleas e]
    PromptPleas66 --> PromptPleas67[Prompt "Pleas e]
    PromptPleas67 --> PromptPleas68[Prompt "Pleas e]
    PromptPleas68 --> PromptPleas69[Prompt "Pleas e]
    PromptPleas69 --> PromptPleas70[Prompt "Pleas e]
    PromptPleas70 --> PromptPleas71[Prompt "Pleas e]
    PromptPleas71 --> PromptPleas72[Prompt "Pleas e]
    PromptPleas72 --> PromptPleas73[Prompt "Pleas e]
    PromptPleas73 --> PromptPleas74[Prompt "Pleas e]
    PromptPleas74 --> PromptPleas75[Prompt "Pleas e]
    PromptPleas75 --> PromptPleas76[Prompt "Pleas e]
    PromptPleas76 --> PromptPleas77[Prompt "Pleas e]
    PromptPleas77 --> PromptPleas78[Prompt "Pleas e]
    PromptPleas78 --> PromptPleas79[Prompt "Pleas e]
    PromptPleas79 --> PromptPleas80[Prompt "Pleas e]
    PromptPleas80 --> PromptPleas81[Prompt "Pleas e]
    PromptPleas81 --> PromptPleas82[Prompt "Pleas e]
    PromptPleas82 --> PromptPleas83[Prompt "Pleas e]
    PromptPleas83 --> PromptPleas84[Prompt "Pleas e]
    PromptPleas84 --> PromptPleas85[Prompt "Pleas e]
    PromptPleas85 --> PromptPleas86[Prompt "Pleas e]
    PromptPleas86 --> PromptPleas87[Prompt "Pleas e]
    PromptPleas87 --> PromptPleas88[Prompt "Pleas e]
    PromptPleas88 --> PromptPleas89[Prompt "Pleas e]
    PromptPleas89 --> PromptPleas90[Prompt "Pleas e]
    PromptPleas90 --> PromptPleas91[Prompt "Pleas e]
    PromptPleas91 --> PromptPleas92[Prompt "Pleas e]
    PromptPleas92 --> PromptPleas93[Prompt "Pleas e]
    PromptPleas93 --> PromptPleas94[Prompt "Pleas e]
    PromptPleas94 --> PromptPleas95[Prompt "Pleas e]
    PromptPleas95 --> PromptPleas96[Prompt "Pleas e]
    PromptPleas96 --> PromptPleas97[Prompt "Pleas e]
    PromptPleas97 --> PromptPleas98[Prompt "Pleas e]
    PromptPleas98 --> PromptPleas99[Prompt "Pleas e]
    PromptPleas99 --> PromptPleas100[Prompt "Pleas e]
    PromptPleas100 --> PromptPleas101[Prompt "Pleas e]
    PromptPleas101 --> PromptPleas102[Prompt "Pleas e]
    PromptPleas102 --> PromptPleas103[Prompt "Pleas e]
    PromptPleas103 --> PromptPleas104[Prompt "Pleas e]
    PromptPleas104 --> PromptPleas105[Prompt "Pleas e]
    PromptPleas105 --> PromptPleas106[Prompt "Pleas e]
    PromptPleas106 --> PromptPleas107[Prompt "Pleas e]
    PromptPleas107 --> PromptPleas108[Prompt "Pleas e]
    PromptPleas108 --> PromptPleas109[Prompt "Pleas e]
    PromptPleas109 --> PromptPleas110[Prompt "Pleas e]
    PromptPleas110 --> PromptPleas111[Prompt "Pleas e]
    PromptPleas111 --> PromptPleas112[Prompt "Pleas e]
    PromptPleas112 --> PromptPleas113[Prompt "Pleas e]
    PromptPleas113 --> PromptPleas114[Prompt "Pleas e]
    PromptPleas114 --> PromptPleas115[Prompt "Pleas e]
    PromptPleas115 --> PromptPleas116[Prompt "Pleas e]
    PromptPleas116 --> PromptPleas117[Prompt "Pleas e]
    PromptPleas117 --> PromptPleas118[Prompt "Pleas e]
    PromptPleas118 --> PromptPleas119[Prompt "Pleas e]
    PromptPleas119 --> PromptPleas120[Prompt "Pleas e]
    PromptPleas120 --> PromptPleas121[Prompt "Pleas e]
    PromptPleas121 --> PromptPleas122[Prompt "Pleas e]
    PromptPleas122 --> PromptPleas123[Prompt "Pleas e]
    PromptPleas123 --> PromptPleas124[Prompt "Pleas e]
    PromptPleas124 --> PromptPleas125[Prompt "Pleas e]
    PromptPleas125 --> PromptPleas126[Prompt "Pleas e]
    PromptPleas126 --> PromptPleas127[Prompt "Pleas e]
    PromptPleas127 --> PromptPleas128[Prompt "Pleas e]
    PromptPleas128 --> PromptPleas129[Prompt "Pleas e]
    PromptPleas129 --> PromptPleas130[Prompt "Pleas e]
    PromptPleas130 --> PromptPleas131[Prompt "Pleas e]
    PromptPleas131 --> PromptPleas132[Prompt "Pleas e]
    PromptPleas132 --> PromptPleas133[Prompt "Pleas e]
    PromptPleas133 --> PromptPleas134[Prompt "Pleas e]
    PromptPleas134 --> PromptPleas135[Prompt "Pleas e]
    PromptPleas135 --> PromptPleas136[Prompt "Pleas e]
   
```

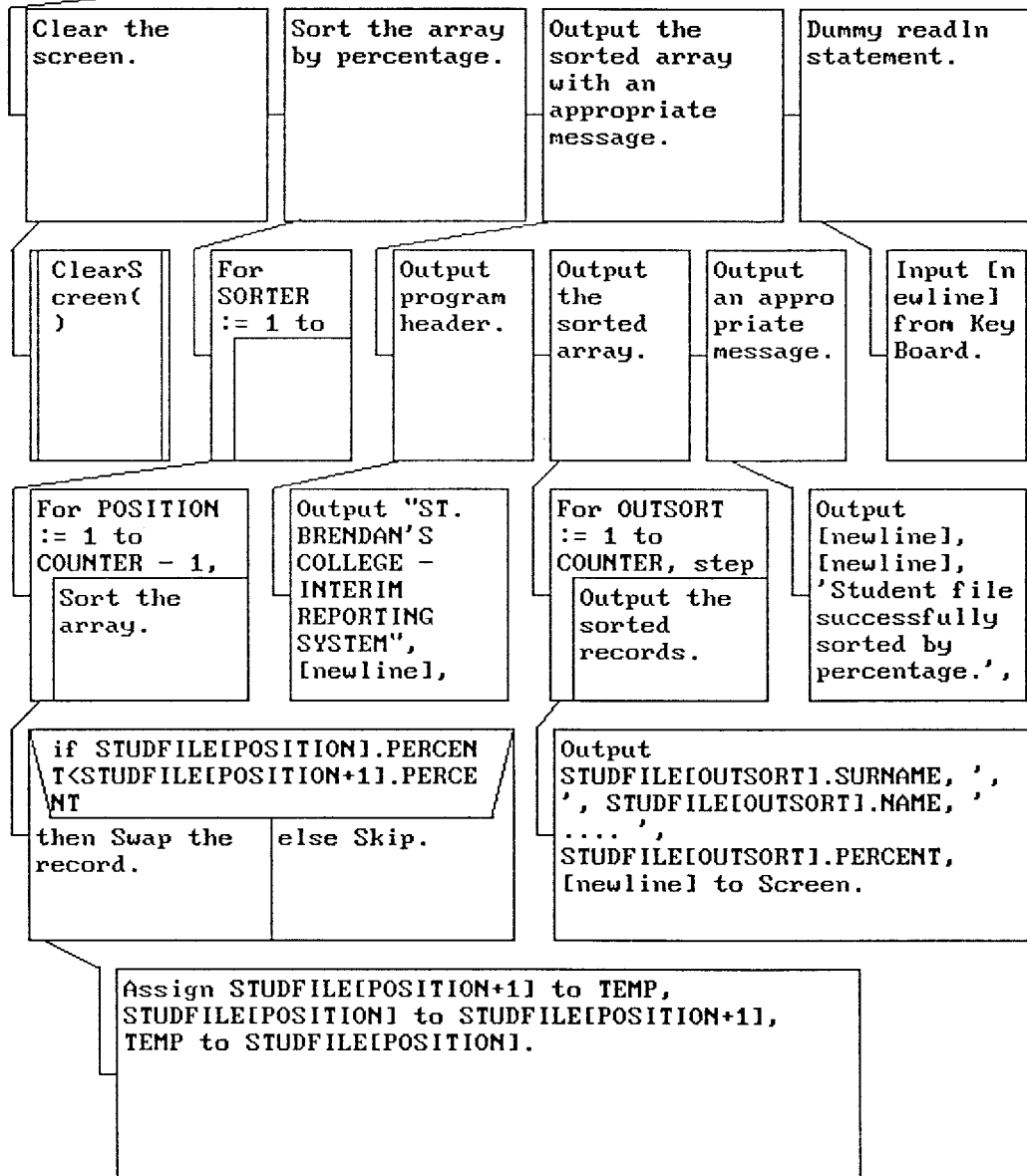
Variable declarations (6)

```

Local PASSWORD : string[10];
Global COUNTER : integer single;
Global STUDFILE : array[1..30] of STUDENT;
Local AGAIN : character;
Global STUDTEXT : textfile;
Local ELEMENT : integer single;

```

This procedure is designed to sort the student array by percentage, by working in pairs - comparing adjacent pairs and swapping them if necessary. This procedure was designed by Simon Wagner for St. Brendan's College.



Identifiers used in procedure SORTPCNT (User Brendans 1 May 1996)

Type declarations (1)

```

Global STUDENT = record
    SURNAME:STR25
    NAME:STR25
    YEAR:STR2
    SUBJCODE:STR2
    TEACHCODE:STR2
    ACHIEVE:STR3
    PERCENT:integer single
    ATTITUDE:character
    CONDUCT:character
    PROGRESS:character
end; record

```

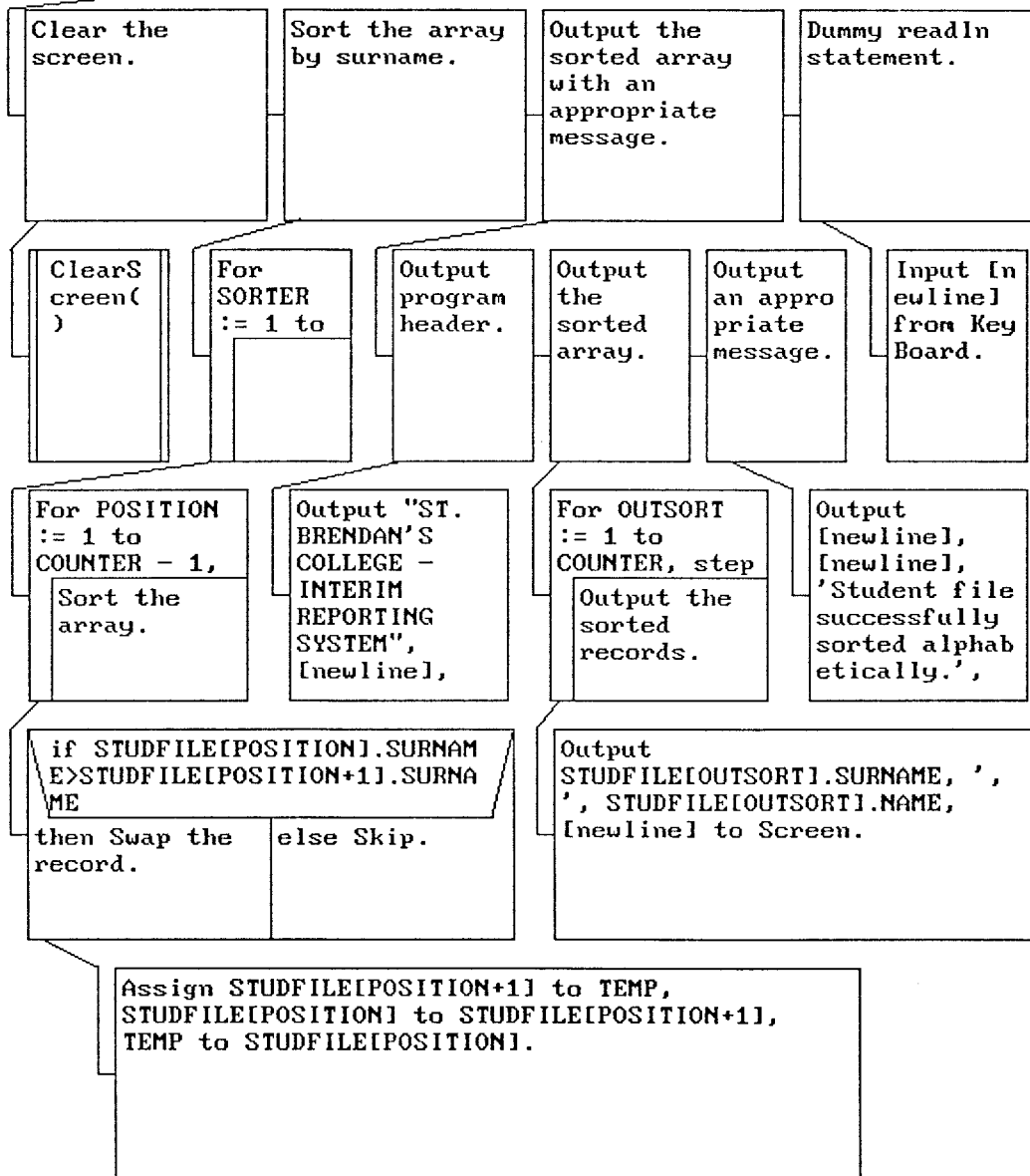
Variable declarations (6)

```

Global STUDFILE : array[1..30] of STUDENT;
Local SORTER : integer single;
Global COUNTER : integer single;
Local POSITION : integer single;
Local TEMP : STUDENT;

```

This procedure is designed to sort the student array alphabetically by name, by working in pairs - comparing adjacent pairs and swapping them if necessary. This procedure was designed by Simon Wagner for St. Brendan's College.



Identifiers used in procedure SORTNAME (User Brendans 1 May 1996)

Type declarations (1)

```

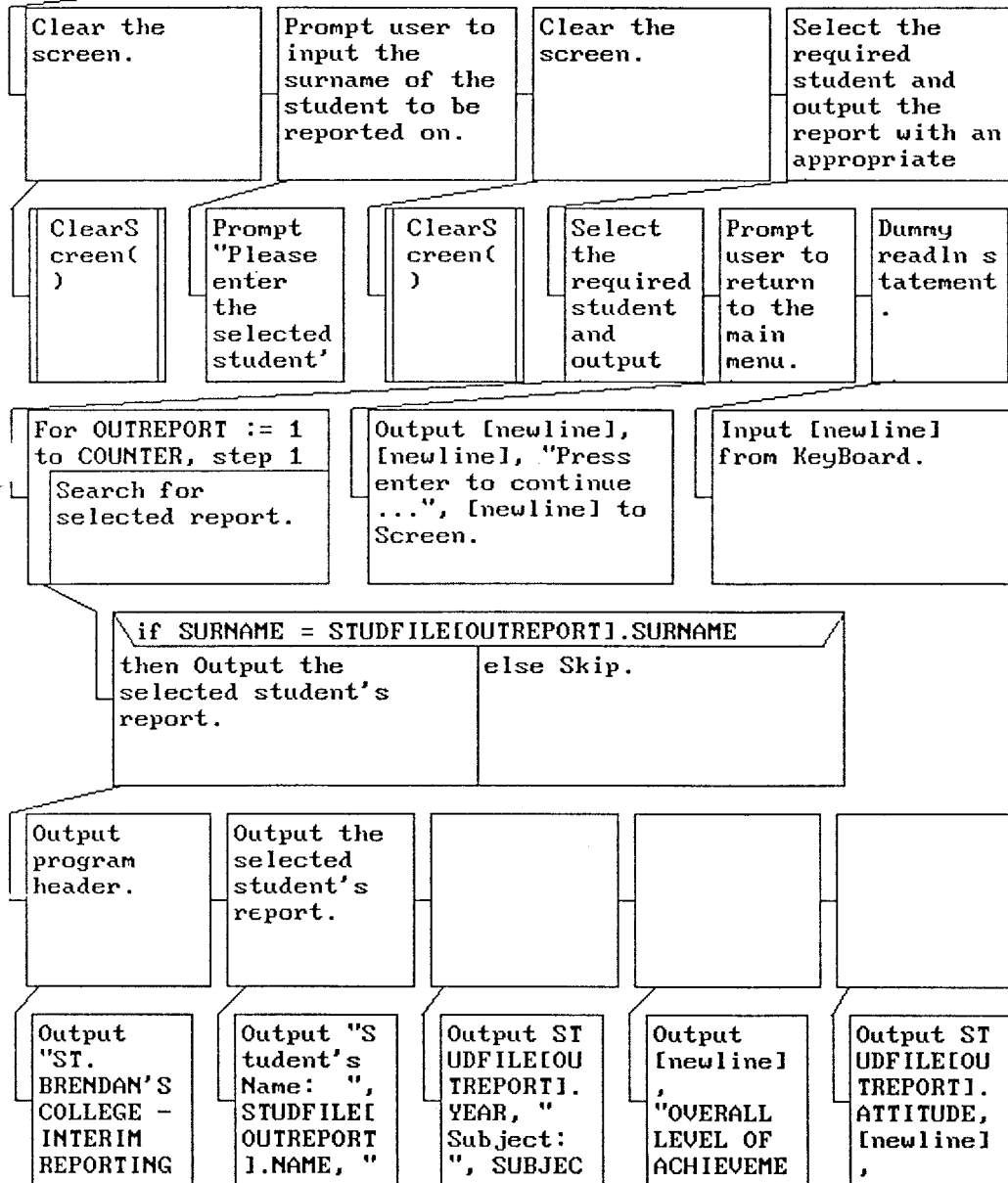
Global STUDENT = record
  SURNAME:STR25
  NAME:STR25
  YEAR:STR2
  SUBJCODE:STR2
  TEACHCODE:STR2
  ACHIEVE:STR3
  PERCENT:integer single
  ATTITUDE:character
  CONDUCT:character
  PROGRESS:character
end; record
  
```

Variable declarations (6)

```

Global STUDFILE : array[1..30] of STUDENT;
Local SORTER : integer single;
Global COUNTER : integer single;
Local POSITION : integer single;
Local TEMP : STUDENT;
  
```

This procedure is designed to output a selected report to the screen. This procedure was designed by Peter Doyle and Lucas Wyte for St. Brendan's College.



Identifiers used in procedure OUTSLCTD (User Brendans 1 May 1996)

Type declarations (1)

Global STR25 = string[25];

Variable declarations (6)

Global STUDFILE : array[1..30] of STUDENT;

Local SURNAME : STR25;

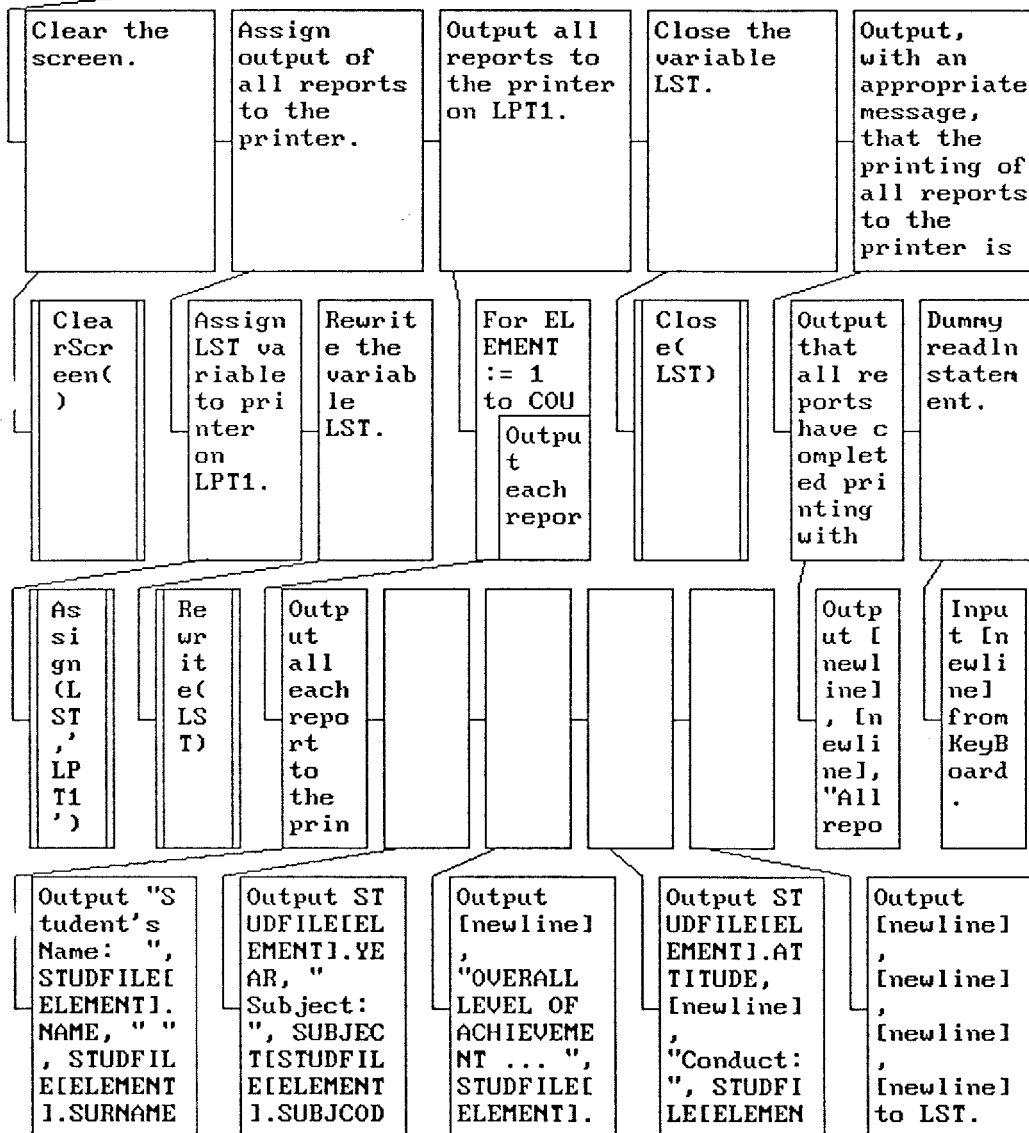
Global SUBJECT : array[1..20] of STR25;

Global TEACHER : array[1..20] of STR25;

Local OUTREPORT : integer single;

Global COUNTER : integer single;

This procedure is designed to output all student reports to the printer on LPT1. This procedure was designed by Peter Doyle and Lucas Wyte for St. Brendan's College.



Identifiers used in procedure OUTALL (User Brendans 1 May 1996)

Variable declarations (7)

Local LST : textfile;

Local COUNT : integer single;

Global COUNTER : integer single;

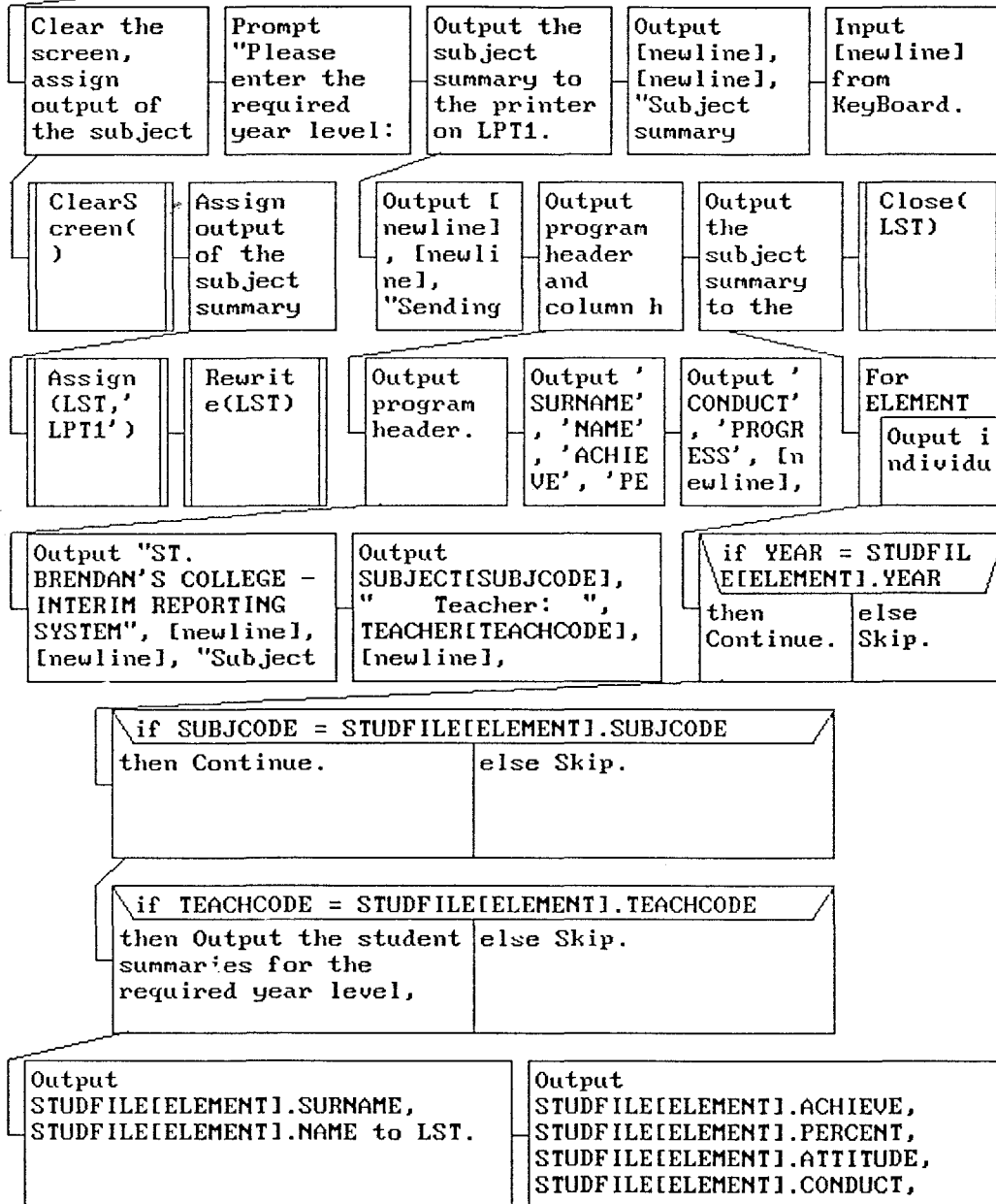
Global TEACHER : array[1..15] of STR25;

Global SUBJECT : array[1..15] of STR25;

Global STUDFILE : array[1..30] of STUDENT;

Local ELEMENT : integer single;

This procedure is designed to output a subject summary to the printer. This procedure was designed by Simon Wagner for St. Brendan's College.



Identifiers used in procedure OUTSUMMY (User Brendans 1 May 1996)

Variable declarations (9)

Local LST : textfile;

Local YEAR : integer single;

Local SUBJCODE : integer single;

Local TEACHCODE : integer single;

Global SUBJECT : array[1..20] of STR25;

Global TEACHER : array[1..20] of STR25;

Local ELEMENT : integer single;

Global COUNTER : integer single;

Global STUDFILE : array[1..30] of STUDENT;

**program REPORTER;**

{This program is designed as an interim reporting system which will advise parents of students performance in a particular subject. The reports will be printed on preprinted stationery. This menu was designed by Lucas Wyte. This program was designed by Peter Doyle, Simon Wagner and Lucas Wyte for St. Brendan's College.}

uses Crt;

type

STR25 = string[25];

STUDENT = record

SURNAME:STR25;

NAME:STR25;

YEAR:integer;

SUBJCODE:integer;

TEACHCODE:integer;

ACHIEVE:char;

PERCENT:integer;

ATTITUDE:char;

CONDUCT:char;

PROGRESS:char;

end {record};

var

SELECTION : char;

COUNTER : integer;

STUDFILE : array[1..30] of STUDENT;

STUDTEXT : text;

TEACHER : array[1..20] of STR25;

SUBJECT : array[1..20] of STR25;

**procedure SORTPCNT;**

{This procedure is designed to sort the student array by percentage, by working in pairs - comparing adjacent pairs and swapping them if necessary. This procedure was designed by Simon Wagner for St. Brendan's College.}

var

SORTER : integer;

POSITION : integer;

TEMP : STUDENT;

OUTSORT : integer;

begin {procedure SORTPCNT}

ClrScr;

for SORTER := 1 to COUNTER do

for POSITION := 1 to COUNTER - 1 do

if

STUDFILE[POSITION].PERCENT<STUDFILE[POSITION+1].PERCENT then

begin

TEMP := STUDFILE[POSITION+1];

STUDFILE[POSITION+1] := STUDFILE[POSITION];

```

        STUDFILE[POSITION] := TEMP
    end
else
    {Skip.};
writeln('ST. BRENDAN"S COLLEGE - INTERIM REPORTING SYSTEM');
writeln;
writeln('Student file:');
writeln;
for OUTSORT := 1 to COUNTER do
    begin
        write(STUDFILE[OUTSORT].SURNAME);
        write(', ');
        write(STUDFILE[OUTSORT].NAME);
        write(' .... ');
        writeln(STUDFILE[OUTSORT].PERCENT)
    end;
writeln;
writeln;
writeln('Student file successfully sorted by percentage. ');
writeln;
writeln('Press enter to continue ...');
readln
end; {procedure SORTPCNT}

```

**procedure SORTNAME;**

{This procedure is designed to sort the student array alphabetically by name, by working in pairs - comparing adjacent pairs and swapping them if necessary. This procedure was designed by Simon Wagner for St. Brendan's College.}

var

```

    SORTER : integer;
    POSITION : integer;
    TEMP : STUDENT;
    OUTSORT : integer;

```

begin {procedure SORTNAME}

ClrScr;

for SORTER := 1 to COUNTER do

for POSITION := 1 to COUNTER - 1 do

if

STUDFILE[POSITION].SURNAME>STUDFILE[POSITION+1].SURNAME then

begin

TEMP := STUDFILE[POSITION+1];

STUDFILE[POSITION+1] := STUDFILE[POSITION];

STUDFILE[POSITION] := TEMP

end

else

{Skip.};



```

writeln('ST. BRENDAN"S COLLEGE - INTERIM REPORTING SYSTEM');
writeln;
writeln('Student file:');
writeln;
for OUTSORT := 1 to COUNTER do
begin
    write(STUDFILE[OUTSORT].SURNAME);
    write(', ');
    writeln(STUDFILE[OUTSORT].NAME)
end;
writeln;
writeln;
writeln('Student file successfully sorted alphabetically.');
```

writeln;

writeln('Press enter to continue ...');

readln

end; {procedure SORTNAME}

**procedure OUTSLCTD;**  
 {This procedure is designed to output a selected report to the screen. This procedure was designed by Peter Doyle and Lucas Wyte for St. Brendan's College.}

var

    SURNAME : STR25;

    OUTREPORT : integer;

begin {procedure OUTSLCTD}

    ClrScr;

    write('Please enter the selected student"s surname: ');

    readln(SURNAME);

    ClrScr;

    for OUTREPORT := 1 to COUNTER do

        if SURNAME = STUDFILE[OUTREPORT].SURNAME then

            begin

                writeln('ST. BRENDAN"S COLLEGE - INTERIM REPORTING SYSTEM');

                writeln;

                writeln('Selected student report:');

                writeln;

                writeln;

                write('Student"s Name: ');

                write(STUDFILE[OUTREPORT].NAME);

                write(' ');

                writeln(STUDFILE[OUTREPORT].SURNAME);

                write('Year Level: ');

                write(STUDFILE[OUTREPORT].YEAR);

                write('            Subject: ');

                writeln(SUBJECT[STUDFILE[OUTREPORT].SUBJCODE]);

                write('Teacher: ');

            end

        end

```

        writeln(TEACHER[STUDFILE[OUTREPORT].TEACHCODE]);
        writeln;
        write('OVERALL LEVEL OF ACHIEVEMENT ... ');
        writeln(STUDFILE[OUTREPORT].ACHIEVE);
        writeln;
        write('Attitude: ');
        writeln(STUDFILE[OUTREPORT].ATTITUDE);
        write('Conduct: ');
        writeln(STUDFILE[OUTREPORT].CONDUCT);
        write('Progress: ');
        writeln(STUDFILE[OUTREPORT].PROGRESS)
    end
else
    {Skip.};
writeln;
writeln;
writeln('Press enter to continue ...');
readln
end; {procedure OUTSLCTD}

```

#### **procedure OUTALL;**

{This procedure is designed to output all student reports to the printer on LPT1. This procedure was designed by Peter Doyle and Lucas Wyte for St. Brendan's College.}

```

var
    LST : text;
    COUNT : integer;
    ELEMENT : integer;
begin {procedure OUTALL}
    ClrScr;
    Assign(LST,'LPT1');
    Rewrite(LST);
    for ELEMENT := 1 to COUNTER do
        begin
            write(LST,'Student"s Name: ');
            write(LST,STUDFILE[ELEMENT].NAME);
            write(LST,' ');
            writeln(LST,STUDFILE[ELEMENT].SURNAME);
            write(LST,'Year Level: ');
            write(LST,STUDFILE[ELEMENT].YEAR);
            write(LST,'      Subject: ');
            writeln(LST,SUBJECT[STUDFILE[ELEMENT].SUBJCODE]);
            write(LST,'Teacher: ');
            writeln(LST,TEACHER[STUDFILE[ELEMENT].TEACHCODE]);
            writeln(LST);
            write(LST,'OVERALL LEVEL OF ACHIEVEMENT ... ');
            writeln(LST,STUDFILE[ELEMENT].ACHIEVE);
            writeln(LST);
        end
    end
end;

```

```

        write(LST,'Attitude: ');
        writeln(LST,STUDFILE[ELEMENT].ATTITUDE);
        write(LST,'Conduct: ');
        writeln(LST,STUDFILE[ELEMENT].CONDUCT);
        write(LST,'Progress: ');
        writeln(LST,STUDFILE[ELEMENT].PROGRESS);
        writeln(LST);
        writeln(LST);
        writeln(LST);
        writeln(LST);
    end;
    Close(LST);
    writeln;
    writeln;
    writeln('All reports completed printing. ');
    writeln;
    writeln('Press enter to continue ... ');
    readln
end; {procedure OUTALL}

```

```

procedure OUTSUMMY;
{This procedure is designed to output a subject summary to
the printer. This procedure was designed by Simon Wagner
for St. Brendan's College.}
var
    LST : text;
    YEAR : integer;
    SUBJCODE : integer;
    TEACHCODE : integer;
    ELEMENT : integer;
begin {procedure OUTSUMMY}
    ClrScr;
    Assign(LST,'LPT1');
    Rewrite(LST);
    write('Please enter the required year level: ');
    readln(YEAR);
    write('Please enter your subject code: ');
    readln(SUBJCODE);
    write('Please enter your teacher code: ');
    readln(TEACHCODE);
    writeln;
    writeln;
    writeln('Sending subject summary to printer on LPT1 ... ');
    writeln(LST,'ST. BRENDAN"S COLLEGE - INTERIM REPORTING
SYSTEM');
    writeln(LST);
    write(LST,'Subject Summary - Year ');
    write(LST,YEAR);

```

```

write(LST,');
write(LST,SUBJECT[SUBJCODE]);
write(LST,' Teacher: ');
writeln(LST,TEACHER[TEACHCODE]);
writeln(LST);
writeln(LST);
write(LST,'SURNAME':11);
write(LST,'NAME':10);
write(LST,'ACHIEVE':10);
write(LST,'PERCENT':10);
write(LST,'ATTITUDE':10);
write(LST,'CONDUCT':10);
writeln(LST,'PROGRESS':10);
write(LST,'-----');
writeln(LST,'-----');
for ELEMENT := 1 to COUNTER do
  if YEAR = STUDFILE[ELEMENT].YEAR then
    if SUBJCODE = STUDFILE[ELEMENT].SUBJCODE then
      if TEACHCODE = STUDFILE[ELEMENT].TEACHCODE then
        begin
          write(LST,STUDFILE[ELEMENT].SURNAME:11);
          write(LST,STUDFILE[ELEMENT].NAME:10);
          write(LST,STUDFILE[ELEMENT].ACHIEVE:10);
          write(LST,STUDFILE[ELEMENT].PERCENT:10);
          write(LST,STUDFILE[ELEMENT].ATTITUDE:10);
          write(LST,STUDFILE[ELEMENT].CONDUCT:10);
          writeln(LST,STUDFILE[ELEMENT].PROGRESS:10)
        end
      else
        {Skip.}
      else
        {Skip.}
      else
        {Skip.};
    Close(LST);
    writeln;
    writeln;
    writeln('Subject summary completed printing. ');
    writeln;
    writeln('Press enter to continue ...');
    readln
  end; {procedure OUTSUMMY}

```

#### **procedure APPEND;**

{This procedure is designed to append the student file saved as STUDENTS.RPT upon verification of a password. This procedure was designed by Peter Doyle for St. Brendan's College.}

```

var
  PASSWORD : string[10];
  AGAIN : char;
  ELEMENT : integer;
begin {procedure APPEND}
  ClrScr;
  write('SECURITY CHECK - Please enter the current password: ');
  readln(PASSWORD);
  if PASSWORD = 'jesusluvsu' then
    begin
      repeat
        ClrScr;
        COUNTER := COUNTER + 1;
        write('Please enter the student"s surname: ');
        readln(STUDFILE[COUNTER].SURNAME);
        write('Please enter the student"s first name: ');
        readln(STUDFILE[COUNTER].NAME);
        repeat
          repeat
            write('Please enter the student"s year level: ');
            readln(STUDFILE[COUNTER].YEAR)
          until STUDFILE[COUNTER].YEAR < 13
        until STUDFILE[COUNTER].YEAR > 7;
        write('Please enter your subject code: ');
        readln(STUDFILE[COUNTER].SUBJCODE);
        write('Please enter your teacher code: ');
        readln(STUDFILE[COUNTER].TEACHCODE);
        repeat
          write('Please enter the student"s level of achievement (A-E): ');
          readln(STUDFILE[COUNTER].ACHIEVE)
        until STUDFILE[COUNTER].ACHIEVE in ['A', 'B', 'C', 'D', 'E'];
        repeat
          write('Please enter a mark for the student"s attitude (G, S, N): ');
          readln(STUDFILE[COUNTER].ATTITUDE)
        until STUDFILE[COUNTER].ATTITUDE in ['G', 'S', 'N'];
        repeat
          write('Please enter a mark for the student"s conduct (G, S, N): ');
          readln(STUDFILE[COUNTER].CONDUCT)
        until STUDFILE[COUNTER].CONDUCT in ['G', 'S', 'N'];
        repeat
          write('Please enter a mark for the student"s progress (G, S, N): ');
          readln(STUDFILE[COUNTER].PROGRESS)
        until STUDFILE[COUNTER].PROGRESS in ['G', 'S', 'N'];
        write('Please enter the student"s overall percentage: ');
        readln(STUDFILE[COUNTER].PERCENT);
        write('Do you wish to enter another student"s details (Y/N)? ');
        readln(AGAIN)
      until AGAIN in ['N', 'n'];
      Assign(STUDTEXT, 'a:\students.rpt');
    end
  end
end

```

```

Rewrite(STUDTEXT);
for ELEMENT := 1 to COUNTER do
begin
    writeln(STUDTEXT,STUDFILE[ELEMENT].SURNAME);
    writeln(STUDTEXT,STUDFILE[ELEMENT].NAME);
    writeln(STUDTEXT,STUDFILE[ELEMENT].YEAR);
    writeln(STUDTEXT,STUDFILE[ELEMENT].SUBJCODE);
    writeln(STUDTEXT,STUDFILE[ELEMENT].TEACHCODE);
    writeln(STUDTEXT,STUDFILE[ELEMENT].ACHIEVE);
    writeln(STUDTEXT,STUDFILE[ELEMENT].ATTITUDE);
    writeln(STUDTEXT,STUDFILE[ELEMENT].CONDUCT);
    writeln(STUDTEXT,STUDFILE[ELEMENT].PROGRESS);
    writeln(STUDTEXT,STUDFILE[ELEMENT].PERCENT)
end;
Close(STUDTEXT);
writeln;
writeln;
writeln('Student file appended.');
```

writeln;

```

write('Press enter to continue ...')
end
else
begin
    writeln('** PASSWORD INCORRECT **');
```

writeln;

```

    writeln('Press enter to return to the main menu ...')
end;
PASSWORD := 'NIL';
readln
end; {procedure APPEND}
```

#### **procedure LOADSTUD;**

{This procedure is designed to load the student file saved as STUDENTS.RPT into memory. This procedure was designed by Peter Doyle and Simon Wagner, and edited by Lucas Wyte for St. Brendan's College.}

```

begin {procedure LOADSTUD}
    Assign(STUDTEXT,'a:\students.rpt');
    Reset(STUDTEXT);
    COUNTER := 0;
    while NOT EOF(STUDTEXT) do
begin
    COUNTER := COUNTER + 1;
    readln(STUDTEXT,STUDFILE[COUNTER].SURNAME);
    readln(STUDTEXT,STUDFILE[COUNTER].NAME);
    readln(STUDTEXT,STUDFILE[COUNTER].YEAR);
    readln(STUDTEXT,STUDFILE[COUNTER].SUBJCODE);
    readln(STUDTEXT,STUDFILE[COUNTER].TEACHCODE);
```

```

        readln(STUDTEXT,STUDFILE[COUNTER].ACHIEVE);
        readln(STUDTEXT,STUDFILE[COUNTER].ATTITUDE);
        readln(STUDTEXT,STUDFILE[COUNTER].CONDUCT);
        readln(STUDTEXT,STUDFILE[COUNTER].PROGRESS);
        readln(STUDTEXT,STUDFILE[COUNTER].PERCENT)
    end;
    Close(STUDTEXT)
end; {procedure LOADSTUD}

```

```

procedure LOADSUBJ;
{This procedure is designed to load the subject file saved as
SUBJECTS.DAT into memory.}
var
    SUBJTEXT : text;
    ELEMENT : integer;
begin {procedure LOADSUBJ}
    Assign(SUBJTEXT,'a:\subjects.dat');
    Reset(SUBJTEXT);
    for ELEMENT := 1 to 20 do
        readln(SUBJTEXT,SUBJECT[ELEMENT]);
    Close(SUBJTEXT)
end; {procedure LOADSUBJ}

```

```

procedure LOADTCH;
{This procedure is designed to load the teacher file saved as
TEACHERS.DAT into memory.}
var
    SUBJTEXT : text;
    ELEMENT : integer;
    TCHTEXT : text;
begin {procedure LOADTCH}
    Assign(TCHTEXT,'a:\teachers.dat');
    Reset(TCHTEXT);
    for ELEMENT := 1 to 20 do
        readln(TCHTEXT,TEACHER[ELEMENT]);
    Close(TCHTEXT)
end; {procedure LOADTCH}

```

```

begin {program REPORTER}
    SELECTION := '0';
    while SELECTION <> '7' do
        begin
            ClrScr;
            LOADSTUD;
            LOADSUBJ;
            LOADTCH;
            writeln('ST. BRENDAN"S COLLEGE - INTERIM REPORTING SYSTEM');

```

```

writeln;
writeln('MAIN MENU');
writeln;
writeln('1. Append Student File');
writeln('2. Sort by Percentage');
writeln('3. Sort by Name');
writeln('4. Output a Selected Report to Screen');
writeln('5. Output All Reports to Printer');
writeln('6. Output Subject Summary to Printer');
writeln('7. Quit');
writeln;
writeln;
write('Please make your selection ...');
repeat
    SELECTION := readkey
until SELECTION in ['1', '2', '3', '4', '5', '6', '7'];
case SELECTION of
    '1' :
        APPEND;
    '2' :
        SORTPCNT;
    '3' :
        SORTNAME;
    '4' :
        OUTSLCTD;
    '5' :
        OUTALL;
    '6' :
        OUTSUMMY;
    '7' :
        {Quit.};
    else
        {Skip.}
end {Case}
end
end. {program REPORTER}

```



ST. BRENDAN'S COLLEGE - INTERIM REPORTING SYSTEM

MAIN MENU

1. Append Student File
2. Sort by Percentage
3. Sort by Name
4. Output a Selected Report to Screen
5. Output All Reports to Printer
6. Output Subject Summary to Printer
7. Quit

Please make your selection ...

ST. BRENDAN'S COLLEGE - INTERIM REPORTING SYSTEM

Student file:

Wyte, Lucas ..... 100  
Hamilton, Ron ..... 98  
Harth, Christopher ..... 88  
Ware, Andrew ..... 82  
Wannai, Isaac ..... 74  
Wagner, Simon ..... 59  
Taylor, Frank ..... 56  
Porter, Ben ..... 38  
Sommerville, Donald ..... 33  
Doyle, Peter ..... 2

Student file successfully sorted by percentage.

Press enter to continue ...

ST. BRENDAN'S COLLEGE - INTERIM REPORTING SYSTEM

Student file:

Doyle, Peter  
Hamilton, Ron  
Harth, Christopher  
Porter, Ben  
Sommerville, Donald  
Taylor, Frank  
Wagner, Simon  
Wannai, Isaac  
Ware, Andrew  
Wyte, Lucas

Student file successfully sorted alphabetically.

Press enter to continue ...

ST. BRENDAN'S COLLEGE - INTERIM REPORTING SYSTEM

Selected student report:

Student's Name: Ron Hamilton  
Year Level: 10 Subject: Accounting  
Teacher: Mr I. Bradshaw

OVERALL LEVEL OF ACHIEVEMENT ... A

Attitude: G  
Conduct: S  
Progress: N

Press enter to continue ...

Student's Name: Ron Hamilton  
Year Level: 10 Subject: Accounting  
Teacher: Mr I. Bradshaw

OVERALL LEVEL OF ACHIEVEMENT ... A

Attitude: G  
Conduct: S  
Progress: N

Student's Name: Andrew Ware  
Year Level: 10 Subject: Accounting  
Teacher: Mr I. Bradshaw

OVERALL LEVEL OF ACHIEVEMENT ... C

Attitude: S  
Conduct: N  
Progress: S

Student's Name: Ben Porter  
Year Level: 10 Subject: Accounting  
Teacher: Mr I. Bradshaw

OVERALL LEVEL OF ACHIEVEMENT ... D

Attitude: S  
Conduct: S  
Progress: N

Student's Name: Frank Taylor  
Year Level: 10 Subject: Accounting  
Teacher: Mr I. Bradshaw

OVERALL LEVEL OF ACHIEVEMENT ... D

Attitude: S  
Conduct: G  
Progress: G

Student's Name: Donald Sommerville  
Year Level: 12 Subject: Accounting  
Teacher: Mr I. Bradshaw

OVERALL LEVEL OF ACHIEVEMENT ... D

Attitude: S  
Conduct: S  
Progress: N

Student's Name: Lucas Wyte  
Year Level: 12 Subject: Biology  
Teacher: Mr W. Laverty

OVERALL LEVEL OF ACHIEVEMENT ... A

Attitude: G  
Conduct: G  
Progress: G

Student's Name: Peter Doyle  
Year Level: 8 Subject: Legal Studies  
Teacher: Mr B. McGregor

OVERALL LEVEL OF ACHIEVEMENT ... E

Attitude: N  
Conduct: S  
Progress: N

Student's Name: Christopher Harth  
Year Level: 9 Subject: Legal Studies  
Teacher: Mr R. Hamilton

OVERALL LEVEL OF ACHIEVEMENT ... C

Attitude: S  
Conduct: S  
Progress: G

Student's Name: Isaac Wannai  
Year Level: 10 Subject: Communication  
Teacher: Mr S. Johnson

OVERALL LEVEL OF ACHIEVEMENT ... B

Attitude: S  
Conduct: N  
Progress: G

Student's Name: Simon Wagner  
Year Level: 10 Subject: Economics  
Teacher: Mr J. Ingram

OVERALL LEVEL OF ACHIEVEMENT ... D

Attitude: S  
Conduct: S  
Progress: N

ST. BRENDAN'S COLLEGE - INTERIM REPORTING SYSTEM

Subject Summary - Year 10 Accounting      Teacher: Mr I. Bradshaw

| SURNAME  | NAME   | ACHIEVE | PERCENT | ATTITUDE | CONDUCT | PROGRESS |
|----------|--------|---------|---------|----------|---------|----------|
| Hamilton | Ron    | A       | 98      | G        | S       | N        |
| Ware     | Andrew | C       | 82      | S        | N       | S        |
| Porter   | Ben    | D       | 38      | S        | S       | N        |
| Taylor   | Frank  | D       | 56      | S        | G       | G        |