Thames Polytechnic

Incorporating Avery Hill College

School of Computing and Information Technology

M E C H S .

A CAL Program.

M E C H S .

A CAL Program.

By : S J Jabbar.

Class : FDYC4.

ACKNOWLEDGEMENTS.

I would like to thank the following for their assistance with this project :

Liz Bacon : for her project supervision.

Frank Martin : for some valuable references.

Thames Polytechnic Computer Services Centre
: for the facilities and services provided.

School of Information Technology Micro lab Technicians
: for the facilities and services.

Thames Polytechnic Library.

Birmingham City Public Lending Library.

Dennis, Roger and Ann
: for their invaluable help during testing.

SUMMARY.

The use of computers in teaching is almost thirty years old. If this present generation was not educated by them then the next will certainly be educated through Computer Aided Learning.

What exactly CAL is and its development up to the present is briefly discussed in the introduction. The variety of CALs in use, and how infact they are used follows this.

After the introduction, the objectives of this project as stated at its conception are laid out. The the next section reports on the implementation details. This includes hardware and software used and the way the software was developed.

Testing of MECHS, the name of the program, was an on going process during development and the results were used to tailor the program to suite user needs. The nature of these tests and the major changes initiated by the test results are outlined in the report.

The conclusions to be drawn, and the recommendations are the final parts of the report.

CONTENTS.

THE REPORT.

1. INTRODUCTION.

1.1 WHAT IS CAL ?

CAL is an acronym, short for Computer Assisted (or Aided) Learning. A similar term CAI or Computer Assisted Instruction is also used but, for the most part, only in the United States. Other related terms are CBL, CBE and CBT (Computer Based Learning/Education and Training). For the purposes of this report the term CAL shall be used.

CAL is based on the formal method of teaching. This approach was first discovered by Socrates, but not for another two thousand years were the possibilities of this method investigated further. In the 1950s the researches of B. F. Skinner at the Harvard university yielded methodical teaching again in the form of Programmed Learning (PL).

Although books were published, methods were devised and even machines built, based on the work of Skinner, by the mid '60s these had largly dissappeared. PL had not died, however, it had undergone a metamorphosis and became computerised, it was now CAL.

1.2 DEVELOPMENT OF CAL.

In 1958, Rath and Anderson, Two IBM personnel wrote the first CAL program. Using an IBM 650 mainframe, the program taught binary arithmetic. It interacted with the student through the console typewriter. Don Bitzer of the University of Illinois made the next important contribution by using what is now called an authoring language. This is a language that allows the CAL to fill in screens of information (known as frames) and not have to be a computer expert.

The next major developments were produced by two private companies as a result of work sponsored by a grant from the US government. Control Data Corporation (CDC) produced a system called PLATO (Programmed Logic for Automatic Teaching Operations) and Mitre Corporation (MC) produced one called TICCIT (Timeshared, Interactive Computer-Controlled Information Television). PLATO used plasma screens which are transparent and allow colour slides to be superimposed upon computer generated graphics. TICCIT used normal television sets. PLATO had an authoring language called TUTOR which allowed the teacher to write material. MC decided to employ a team of writers instead.

During this time IBM were devloping their own CAL systems. The earlier ones were far too expensive to be developed beyond the research stage but other CAL systems and authoring languages

were spawned such as Coursewriter-III and IIS. Other manufacturers who worked on CAL were Bell and Howell, Texas Instruments and Hewlett-Packard. Specialist CAL companies were founded such as Computer Curriculum Corperation (CCC).

These developments have for the most part been on large mainframes and minis, which meant that they were out of the reach of most schools and colleges never mind an individuals. But since the late '70s, with the widespread use of home computers such as Apple, Tandy, Commodore etc, CAL systems have been implemented on these, much more inexpensive, machines. With the launch of the IBM PC, in 1982, a new era in CAL programs development emerged. Hundreds, perhaps even thousands of CAL systems exist now for microcomputers. Some of the more noteworthy systems are MICROTEXT, PASS, MCSD, MUMEDALA and CYCLOPS. This means that CAL is now within reach of most people with home-computers, and these days most are well within the reach of average households.

1.3 USING A CAL.

A CAL program presents its material to the student using the television screen or a VDU (visual display unit). The usual way by which the student communicates with the CAL program is via a normal typewriter style keyboard. The student is presented with educational material in a structured way and the computer allows him to respond individually, actively, and at his own pace.

Finally the computer is instructed to give accurate and immediate feedback to the student.

1.4 VARIETIES OF CAL.

There are a great variety of educational programs which are loosely described as CAL. Among these are drill and test programs, educational games, simulations, multi-choice tests (with or without accompanying text) and others which can not easily be placed in any particular category.

1.4.1 Drill and test.

This sort of program is suited for infants and juniors, that is for groups of children to have fun together in drills of varius sorts. But generally they are not recommended fo individual use.

1.4.2 Educational games.

These sorts of programs are designed for young children and have similarities to arcade games. Games programs can be used in two ways :

- Firstly, they can be used as rewarding interludes between periods of more formal learning.

Secondly, the whole educational program can be designed as a game.

1.4.3 Simulations.

There are two sorts of simulations - a short-running animated example within a normal CAL or they can be written as complete simulation packages. The latter can be run in real-time or fast time.

In real-time you can learn how to fly an aeroplane or control an industrial process.

In fast time programs, events are condensed into short time periods for the learner, eg. you can be Chancellor of the Exchequer and try to control balance of payment deficits over a long period of years.

1.4.5 Multi-choice tests.

This project is an example of a CAL in this category. It presents text after which the pupil is given questions with a choice of answers. Multi-choice programs are useful for revision purposes.

1.5 CAL FOR PHYSICS.

Physics is a suitable subject for encoding into a CAL as any questions set can usually have but one answer only. Most science subjects are similarly suited for CAL.

On the Open University course 'S271 - Discovering Physics' an important element of the summer school is the use of programs. One of these is a CAL program and runs on a TERAK 8510:a microcomputer. The programs were written in UCSD pascal with the aid of the OASIS authoring sytem.

1.6 PASCAL AS A MEDIUM FOR CAL.

There is no one particular language that is ideally suited for CAL. Indeed nearly every major language in use today, and quite a few minor ones, have been used to write CAL programs. A number of factors have to be taken into consideration when choosing a language to work with. These include : price, availiability, related tools, utilization rate, ease of learning, productivity and special features.

- **BASIC** is the first choice for many who are involved in CAL but that is as a result of its low cost and wide availiability. Its inherent unstructured nature made it unsuitable for this project.

- C would be a better language for a serious program and the C kernal is quite small and fairly easy to master. Although there are extensive libraries in C that make CAL possible, access to a unix machine is vital. The lack of access at the start of this project was, together with the authors lack of experience in C ruled this language out.

- NATAL is the only high-level portable language specially designed for CAL. Unfortunately it is difficult to obtain a commercial version of this language despite micro-NATAL having been around for microcomputers since 1983.

- PASCAL is the language of implementation for this project. his is due to its emphasis on structured programming, its availiability and the authors familiarity with the language.

2. OBJECTIVES.

This project aims to produce a Multi-choice CAL program with text that serves as a revision aid for classical mechanics. It is based upon a current O-level syllabus but should be used by those who have been on a physics course upto O-level standard. However an intelligent non-physics student should be able to use it aswell.

The program consists of six 'chapters' each with a tutorial followed by questions on the text presented. Should the student fail to answer a question correctly a reinforcement frame of text will be presented, explaining the correct answer. At the end of a session the program will give marks for the chapters attempted and also a mark on the overall performance of the student.

The program will be tested during its development by students and the results of these tests will be used to tailor the program accordingly. The students will be both 'physics literate' as well as non-physics users.

3. IMPLEMENTATION DETAILS.

3.1 HARDWARE.

The program MECHS was written on the PRIME 750 minicomputer. The terminal used in developing the program (and consequently the one that it has to be ran on) was the GT100.

3.2 SOFTWARE USED.

MECHS was written using the Sheffield screen editor SED under the primos operating system. The language for program development was Sheffield pascal.

3.3 SOFTWARE DEVELOPMENT.

The software was developed using structured techniques.

3.3.1 Functional specification.

Produce a physics revision aid CAL program that will do the following :

a) present text via frames,

b) test the student on the text presented,

c) receive and analyse student responses,

d) provide reinforcement material if required.

The subject matter of this CAL program should be classical mechanics and should be aimed mostly at students with ordinary level physics knowledge.

3.3.2 Data Flow Diagrams.

SEE APPENDIX 3.1

3.3.3 Structure Charts.

SEE APPENDIX 3.2

4. TESTING.

The best method for testing a CAL is to give it to a potential user and see how he/she gets on with it.

The tests were carried out after the program was written in skeletal form i.e without page numbering or text justification and without the full range of chapters.

TEST No. 1; (by DENNIS - 2nd year computing science student).

Being a computing science student he found no difficulty in getting to the system and running MECHS. His observations were :

"The introduction is too short and not entirely relevent to what happens after."

- The introduction was lengthened and was updated to include information about the changes that had been made since it was first written.

"It's confusing to read, especially when text is split across two pages and the message about pressing any key is in between."

- The text was rewritten so as not to start a major sub-section on the bottom of a screen.

TEST No. 2; (by ROGER - a first year engineering student.)

Again there were no major difficulties in getting to MECHS. His observations were :

"It would have been nice if the pages were numbered and the screen was split up at the top and bottom."

- Coding that allowed numbering of both the frames of text and the question frames was added. Using the GT100s alternative character set, a routine was also included to split the screen into three parts; one for the page number, one for the text and one for the command line at the bottom.

TEST No. 3; (by ANN - a first year biology student.)

Login proved to be difficult but after a rough user guide had been supplied, MECHS was initiated easily. Her observations were :

"There should be some diagrams along with the text."

- This was immpossible to do. See RECOMMENDATIONS.

"The questions are a bit wordy and some of the multiple choice answers are a bit silly."

- The questions that proved too wordy were shortened and the silly multiple choice answers were replaced by more probable ones.

"The title of the chapter thats being done should be on the screen somewhere."

- Coding to put the title on the screen was added. The title appears on the same line as the page number.

"The diagnostics part could be more meaningful."

- Instead of the chapters being called chapter 1, chapter 2 etc, The routine to produce the title of the chapter was used to make the diagnostics section more meaningful.

The tests highlighted other problems but these were mostly details. The varius range, type and size errors were encountered by all the test subjects so these were noted and the program can now detect and deal with these.

5. CONCLUSION.

As a revision aid MECHS has proved to be a good system. Students who have not studied physics for a long time say that it served as a 'memory jerker'. The results of MECHS use by non physics students highlighted its inapropriateness as a teaching aid for classical mechanics. But nearly all of these thought it served as a useful introduction to the subject.

The program itself was as efficient as the author could make it while allowing it to be legible for another pascal programmer. There maybe a case for some form of data compression on the text files.

Pascal has been a good medium for writing a CAL, allowing a working version of the program to have been produced at an early stage in the development. Because of pascals structured nature and through the use of structured development techniques work, on MECHS, proved to be wholly compatible with the prototype.

Ordinary level Physics in general, and classical mechanics in particular, is a good subject for CAL. An understanding of the subject matter to a depth greater than needed for O level was required. However, the insight into 're-learning' thus gained, as well as the process by which this understanding was acquired, were an invaluable asset in developing MECHS.

Without the feedback that users provided, during development, the MECHS program would not have been able to meet the design objectives. This proved to be an essential part in the development process. The final product met with approval by most of the users, They were satisfied that it was an aid to revision.

The implementation of MECHS on the PRIME was more than adequate allied, as it was, with the facilites provided by the GT100 terminal. There is scope for future development, on the IBM PC for example. Or by adding more chapters so that a greater area of mechnics may be covered.

MECHS has met its design objectives and it is consistant with current CAL systems design.

RECOMMENDATIONS.

The main recommendation is that MECHS would be well suited to running on the IBM PC. This would allow the adding on of a graphics capability and would give it the portability that a CAL system should aim for.

Currently the MECHS program is linked exclusively to the classical mechanics, the subject of implementation. If the titles, chapter-headings and menu parts of MECHS were made subject independent then that would be a step in the direction of a CAL Revision Shell. Where the subject matter would be written using an authoring language and MECHS could then, easily serve as a front end to a much larger, and more complete, CAL system.

REFERENCES.

1. A. F. ABBOTT (1978). Ordinary Level Physics (3rd Edition), (Heinemann Educational Books).

2. B. F. SKINNER (1954). The science of learning and the art of teaching, (Harvard Educational Review).

3. B. F. SKINNER (1965). Waldon Two, (Macmillan London).

4. E. J. PERKINS (1979). Test Yourself Physics, (Celtic (Revision Aids).

5. J. R. HARTLEY (1978). Computer Assisted Learning, (Computer Based learning Unit Leeds University).

BIBLIOGRAPHY.

1. R. C. ATKINSON, H. A. WILSON - Computer Assisted Instruction. (Academic Press).

2. P. BARKER, H. YEATES - Introducing Computer Assisted Learning. (Prentice/Hall International).

3. G. BEACH - Computer Based Learning. (Sigma Technical Press).

4. A. BORK - Learning With Computers. (Digital Press).

5. D. GODFREY, S. STERLING - The Elements of CAL. (Reston Publishing Company Incorperated).

6. R. HOOPER, I. TOYE (ed) - Computer Assisted Learning in the United Kingdom. (Councils & Education Press Ltd.).

7. T. O'SHEA, J. SELF - Learning and Teaching with Computers. (Harvester Press).

THE APPENDIX.

APPENDIX I.

1.USER GUIDE.

1.1 PURPOSE OF THE PROGRAM MECHS.

This is a Computer Aided Learning program called MECHS. It is meant to be used as a revision aid for an ordinary level physics course on some aspects of classical mechanics. MECHS will present you with some text, after which you will be asked to answer some questions. The questions will take the form of multiple choice type with correct answers allowing progression to the next question. An incorrect answer prompts the program into presenting some reinforcement text before progression. This pattern is repeated for all the chapters that you decide to attempt. At the end of the session you will be given a breakdown of the marks you scored at each section.

1.2 DESCRIPTIONS OF MECHS's FACILITIES.

1.2.1 Introduction.

If requested a few introductory frames of text will be displayed. Outlining the objectives and function of the program.

1.2.2 Menu.

The menu provides a list of your options at that particular stage, these are :

1. Force of Gravity, Weight & Friction.
2. Speed, Velocity & Acceleration.
3. Newtons Laws of Motion.
4. Work, Energy & Power.
5. Machines.
6. Density and Relative Density.

(Each of these chapters follows the pattern outlined earlier on in this guide.)

7. End This Session.

(If this option is chosen then the program will produce some diagnostics. It will indicate whether a particular chapter was attempted by displaying the chapter name along with the mark scored on the questions. If a chapter was not attempted then this is indicated aswell. Finally, an overall mark is produced before MECHS ends.

1.3 RESOURCES REQUIRED.

The MECHS program runs on a PRIME 750 minicomputer and requires 8 pages of store. It was written in Sheffield pascal. The text files require a minimum of 15 pages of store. A typical session will need between twenty and forty minutes of run time to complete. The GT100 terminal is the only one that MECHS will run on at the time of writing.

1.4 TO RUN MECHS.

If you have an account on the Prime then to run the program you must :

First LOGIN to your UFD (N.B must be through a GT100).

Then type SEG <TPB2>JS1CY4>PROJECT>MECHS.SEG

If you have not got an account then find a GT100 terminal and do the following:

Switch the terminal on,

press the SHIFT and BREAK keys together and wait for FACILITY REQUIRED ? to appear. You should type CO1 and wait for the message [CONNECTED]. When it does type LOGIN JS1CY4 you will then be asked for the Password : you should type COPPED (the password will not appear on screen). You will now have logged in. As soon as the OK, prompt appears you may begin the session by typing SEG PROJECT>MECHS .

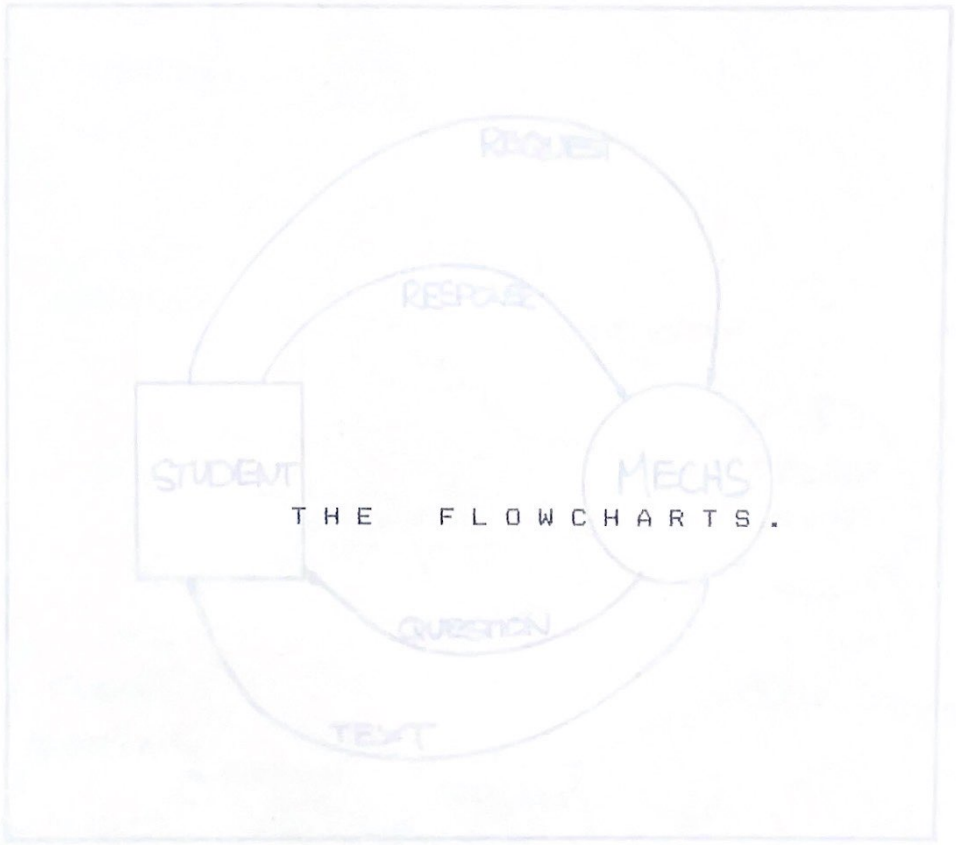
When you have finished the session type LOGOUT to leave the UFD.

EXAMPLE RUN SESSION

Example run sessions have, unfortunately, not been able to be included. This is as a result of there not being any easy way of getting what appears on the screen of the GT100 on to paper.

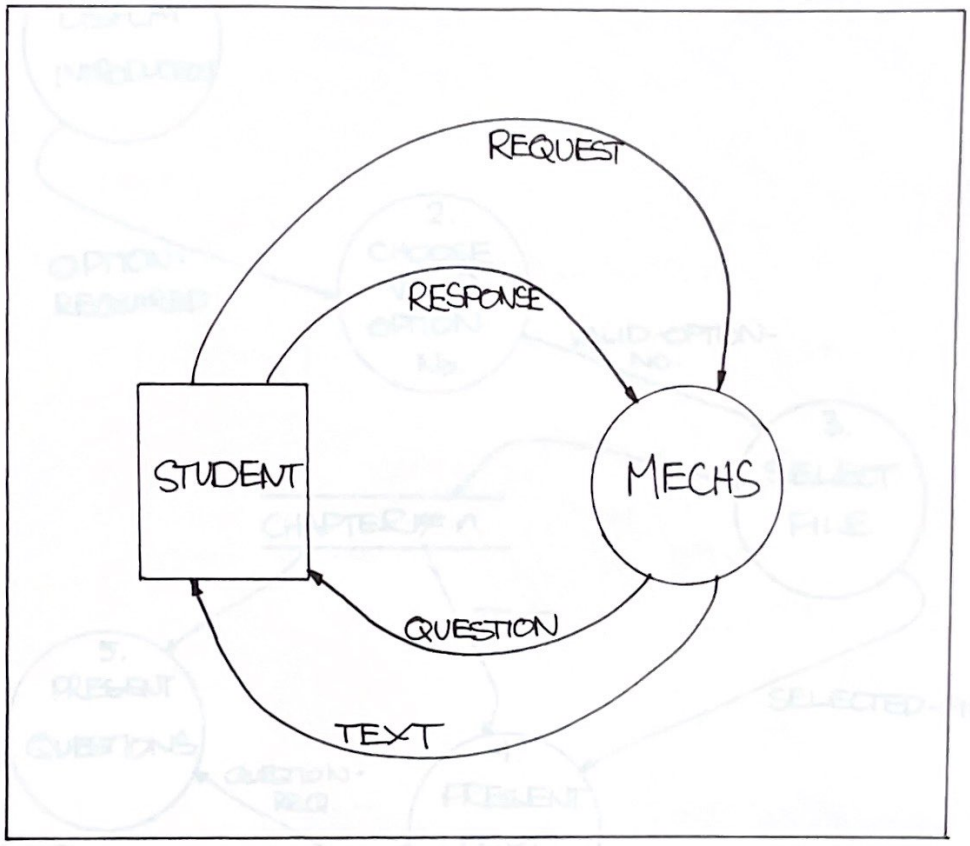
However, if you want to see a run session without actually running MECHS then this is possible. First login to the UFD as described in the USER GUIDE, then instead of running MECHS using the SEG command, type PRINT EXAMPLE.

Logout as before.



CONTEXT DIAGRAM.

THE FLOWCHARTS.



CONTEXT DIAGRAM.

DIAGRAM Ø

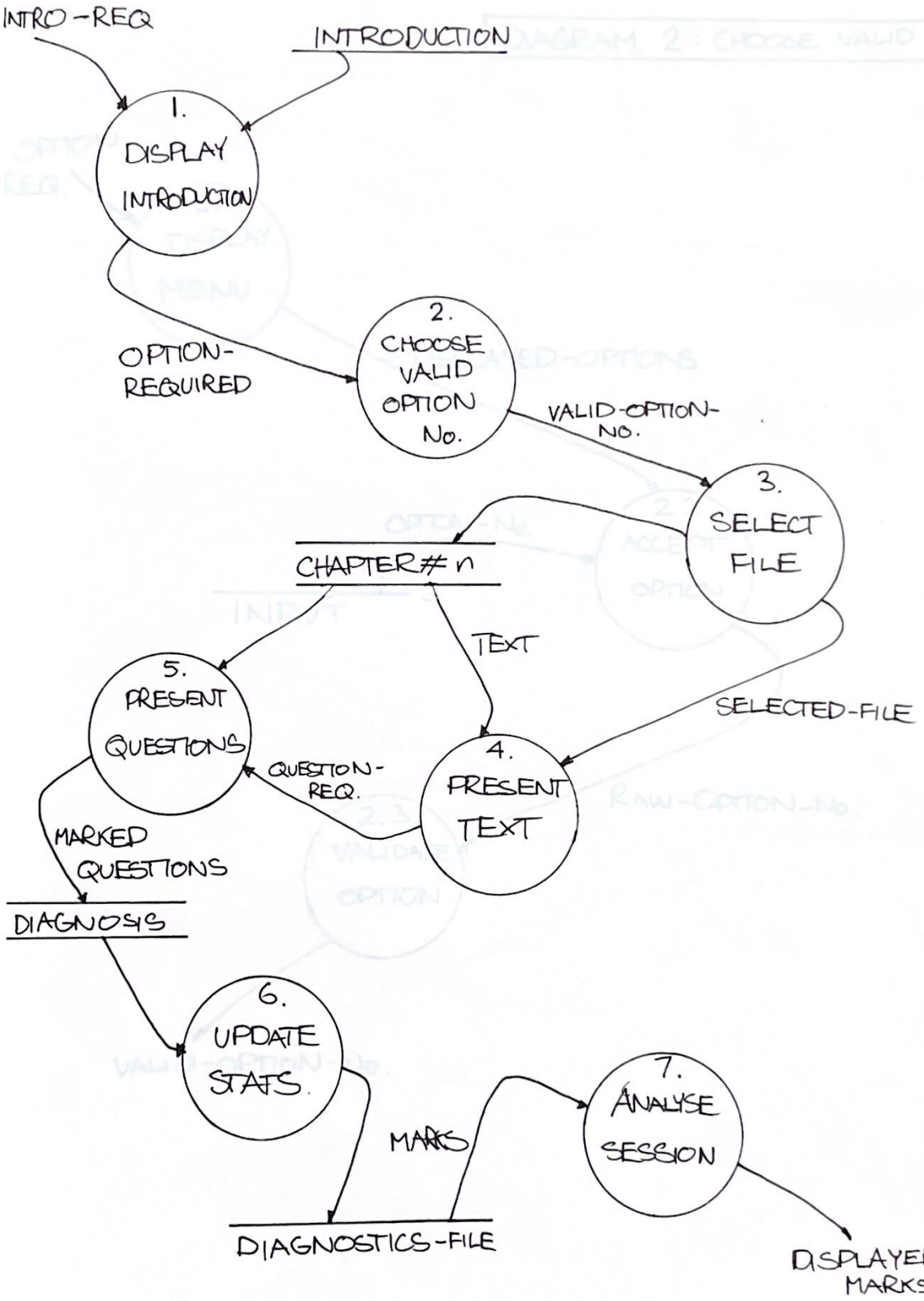

INTRO-REQ
INTRODUCTION
1. DISPLAY INTRODUCTION
OPTION-REQUIRED
2. CHOOSE VALID OPTION No.
VALID-OPTION-NO.
3. SELECT FILE
CHAPTER#n
TEXT
SELECTED-FILE
5. PRESENT QUESTIONS
QUESTION-REQ.
4. PRESENT TEXT
MARKED QUESTIONS
DIAGNOSIS
6. UPDATE STATS.
MARKS
7. ANALYSE SESSION
DIAGNOSTICS-FILE
DISPLAYED-MARKS

DIAGRAM 2 : CHOOSE VALID OPTION

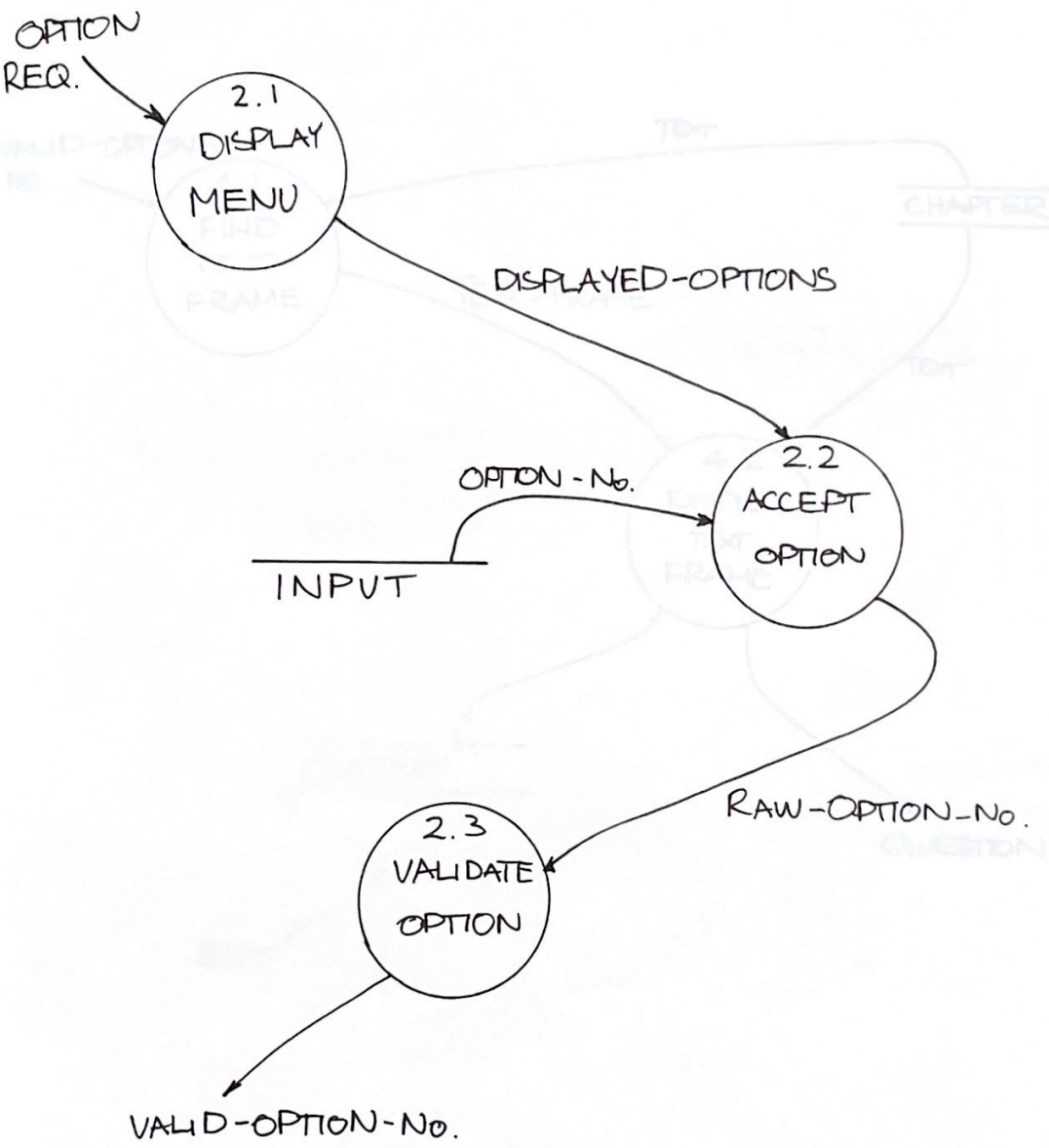


DIAGRAM 4 : DISPLAY-TEXT

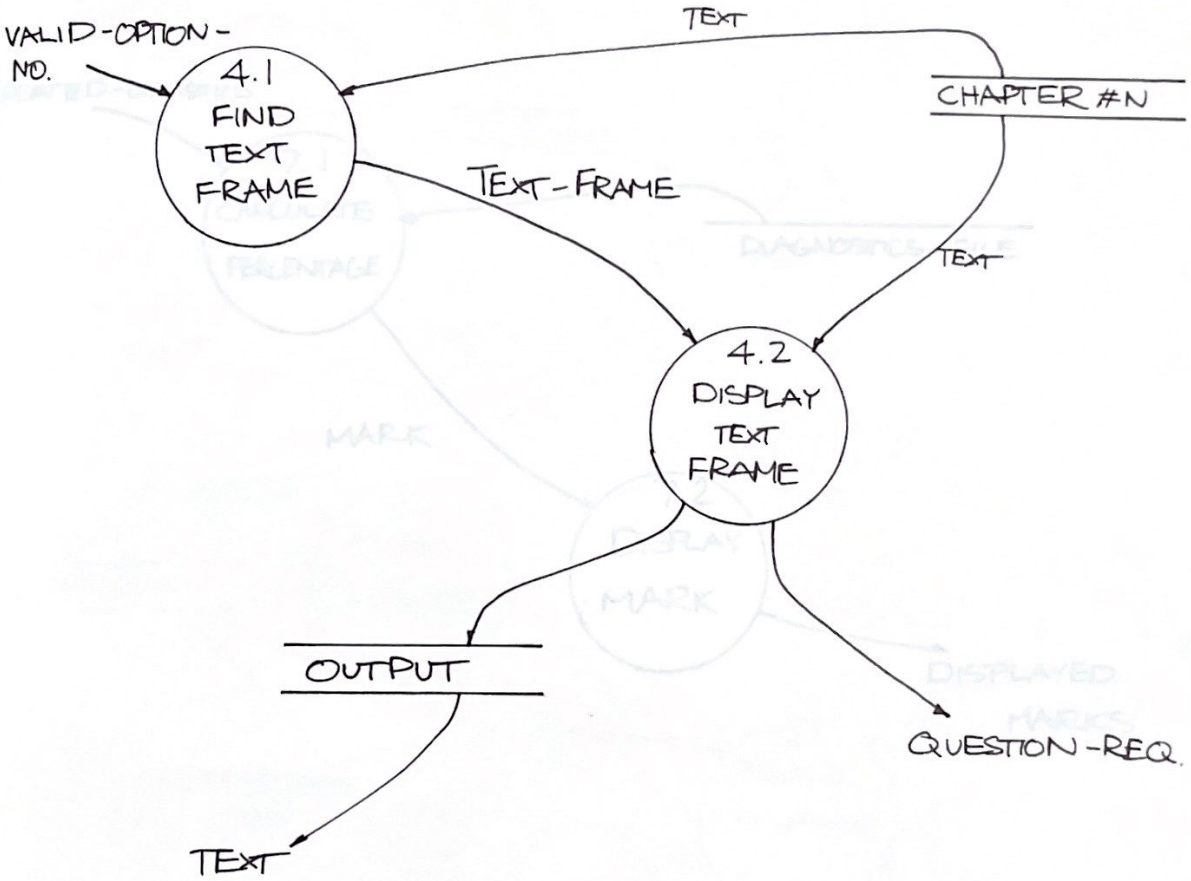


DIAGRAM 7 : ANALYSE SESSION

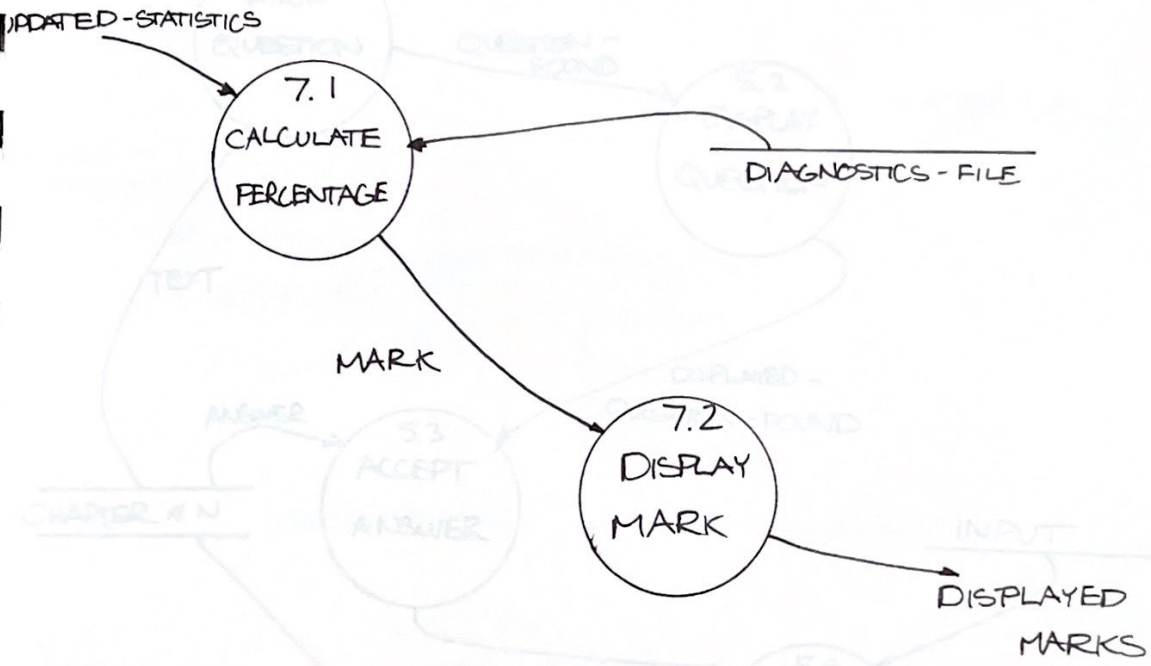


DIAGRAM 5 : PRESENT QUESTIONS

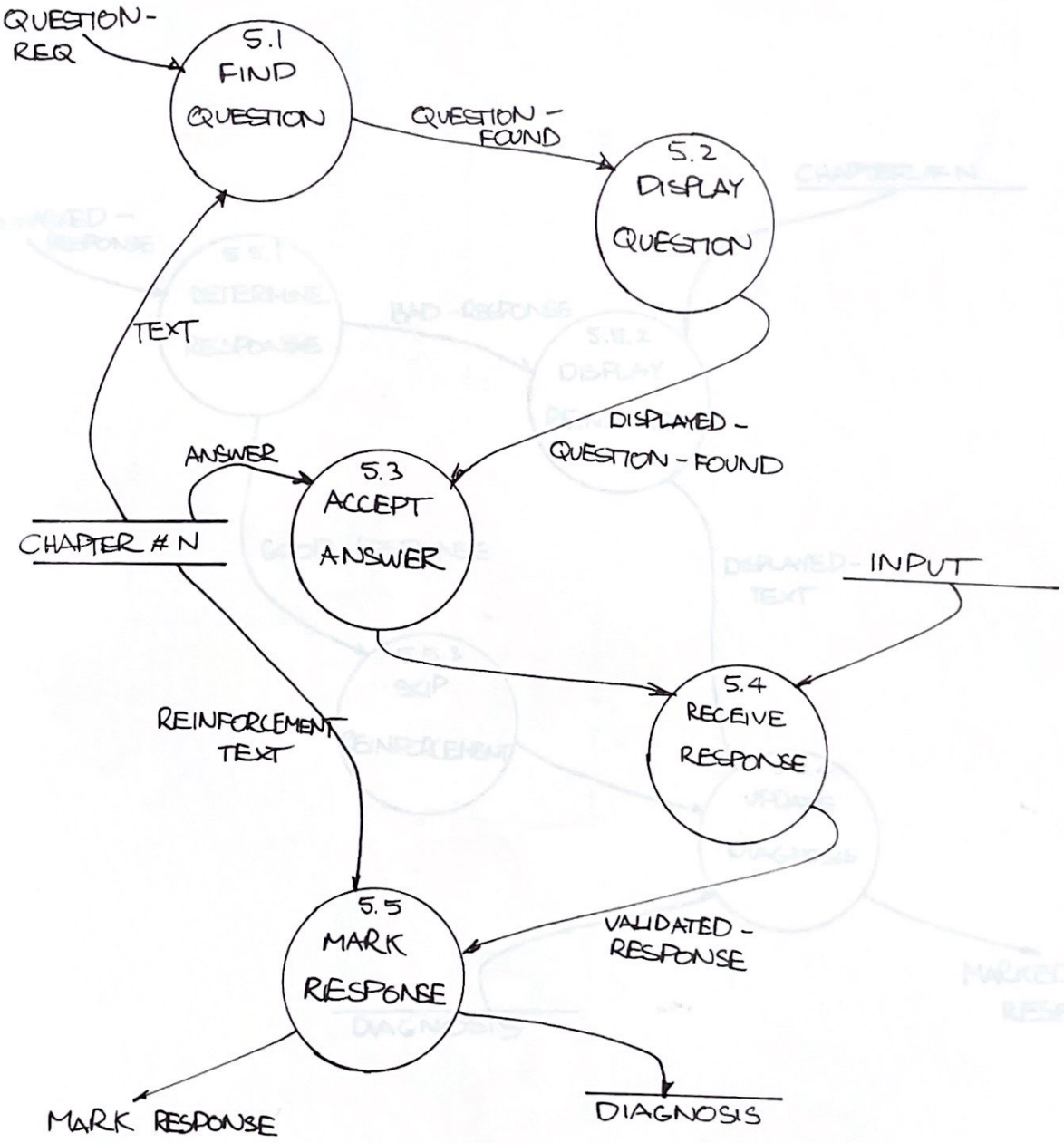
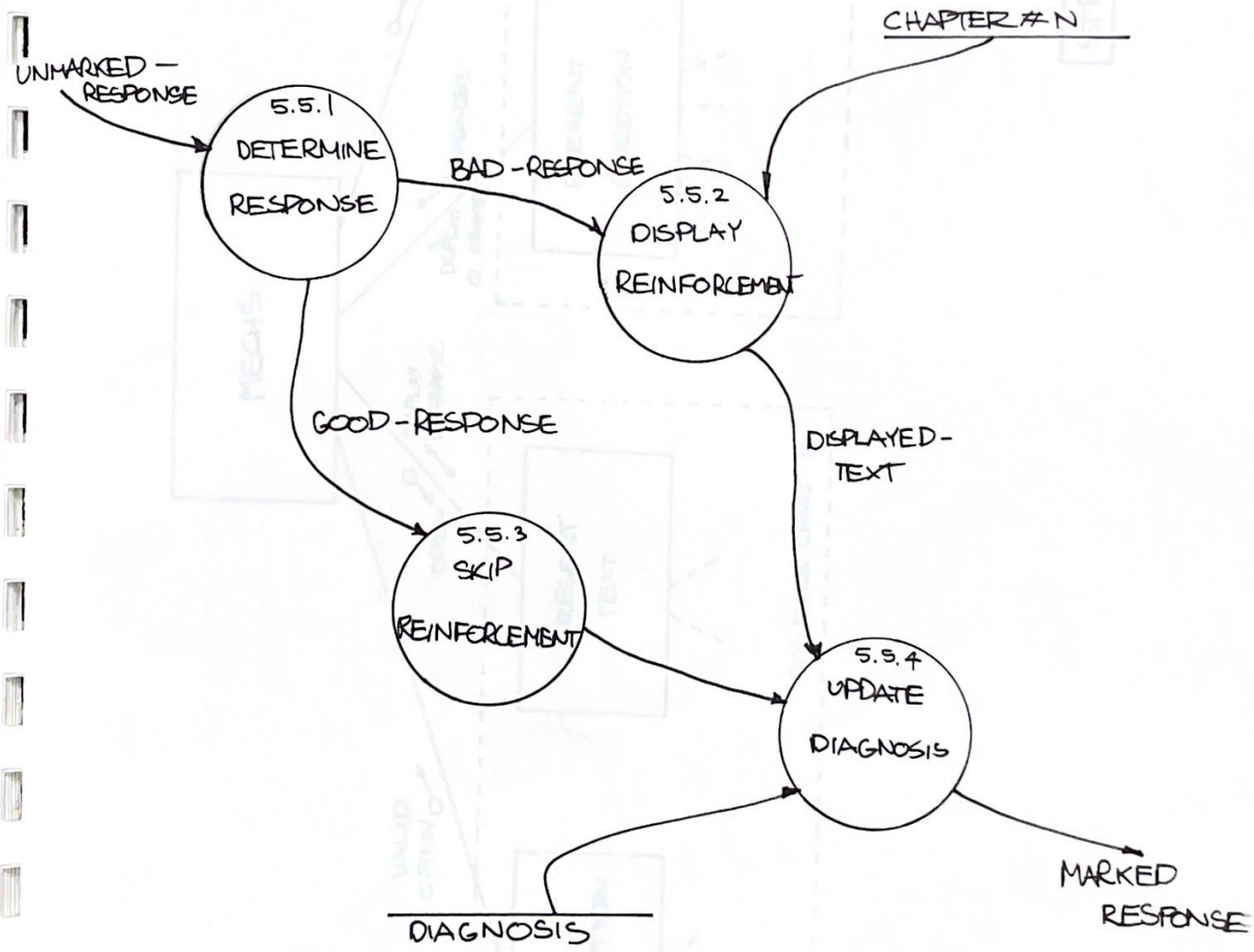
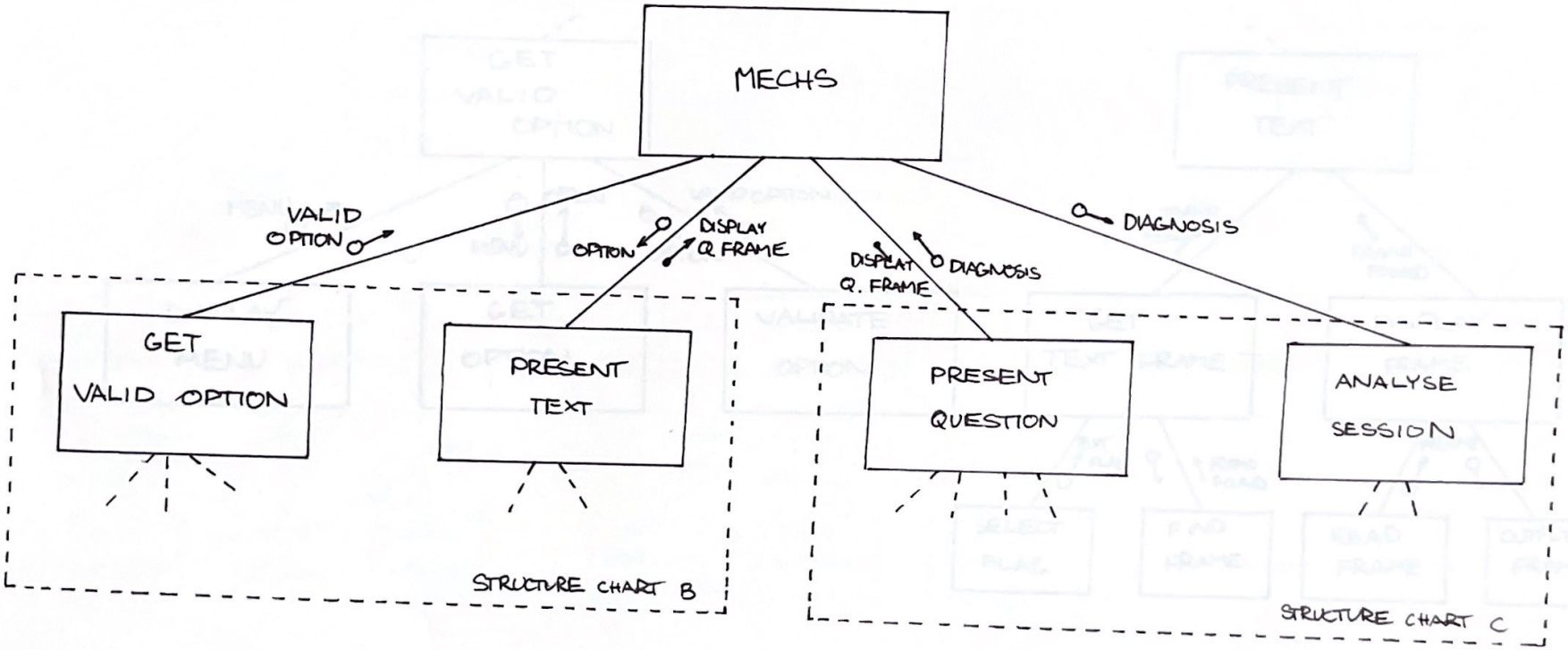
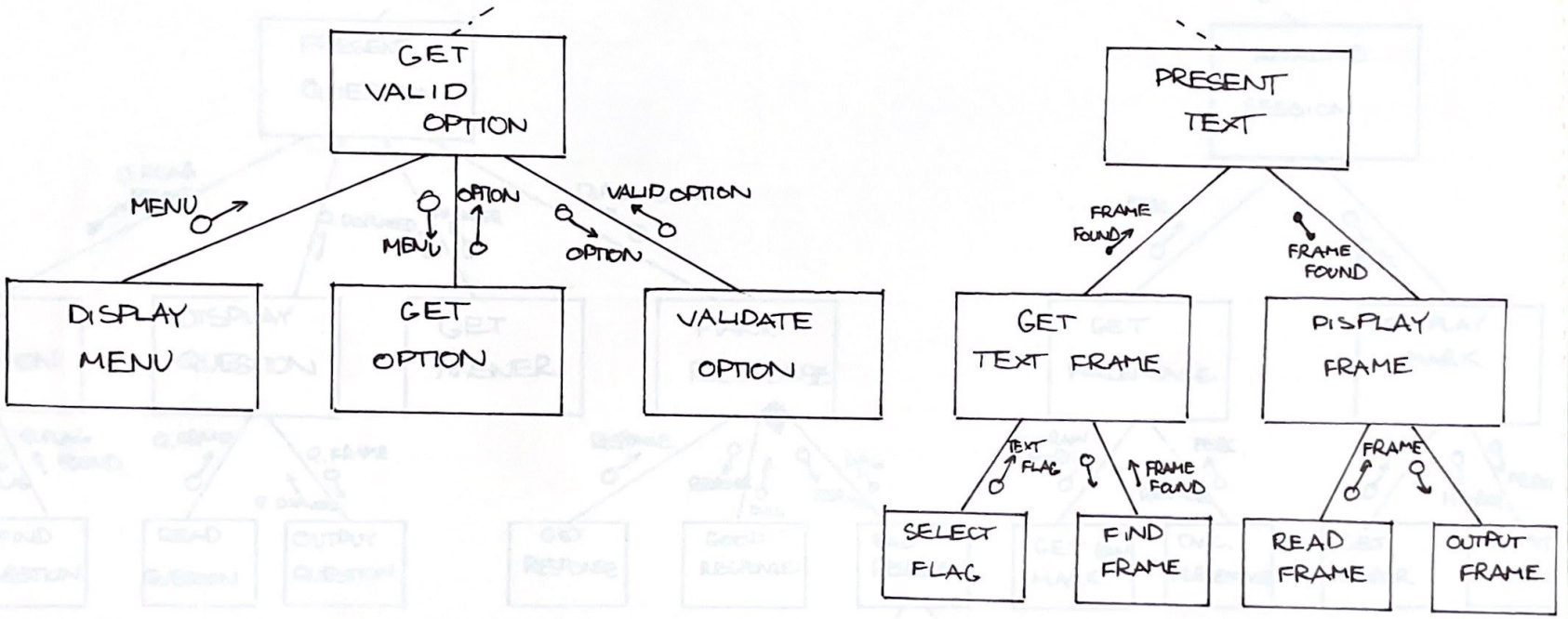


DIAGRAM 5.5 : MARK RESPONSE

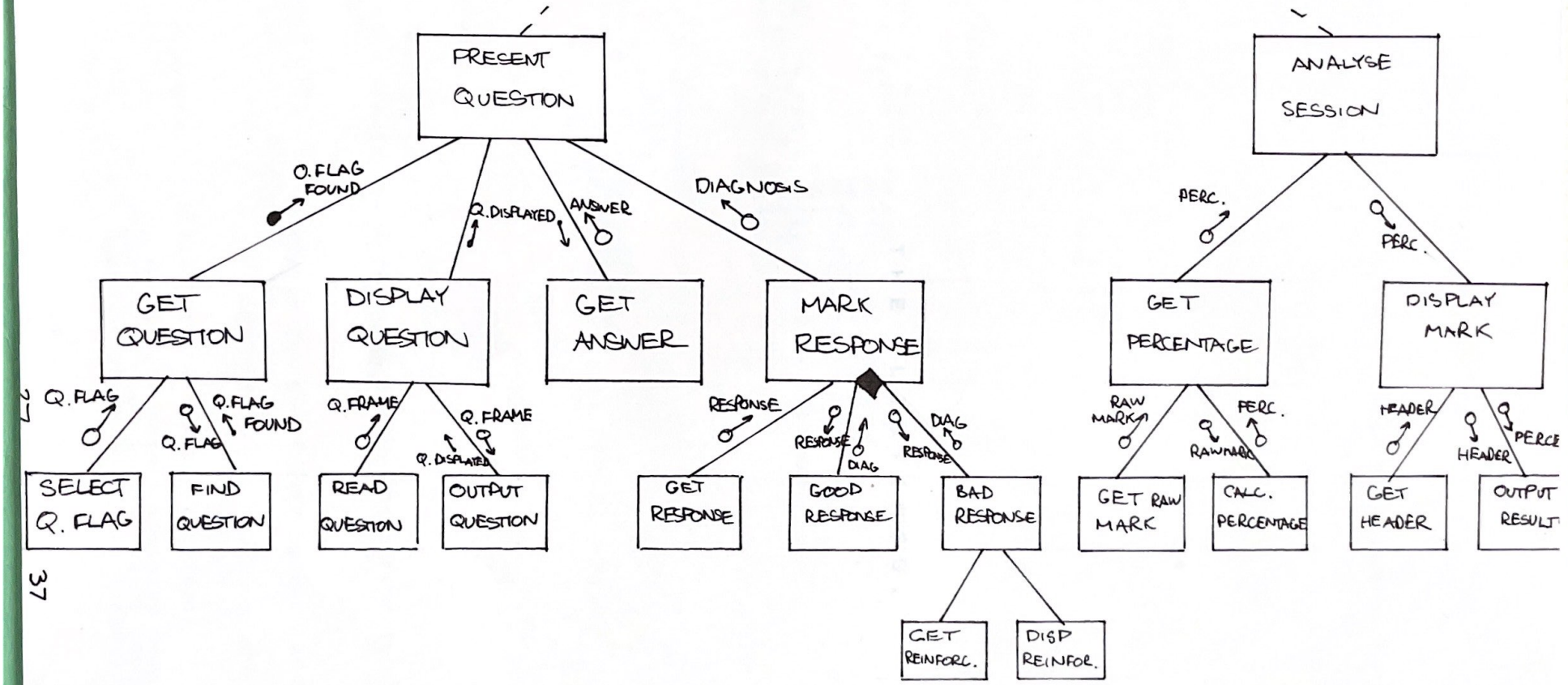




STRUCTURE CHART A.



STRUCTURE CHART B



STRUCTURE CHART C

THE LISTINGS.

```
program mechs;

{The program for the CAL project to teach 'O' level mechanics.
 written   : February - April 1987.
 Author    : S J Jabbar.                                      }

{constants}

const
    null = 0;
    finish = 7;
    text_flag = '{-T}';
    question_flag = '{-Q}';
    answer_flag = '{-A}';
    reinforcement_flag = '{-R}';
    end_of_chapter_flag = '{-E}';
    introduction_flag = '{-I}';



{declarations}

type
    options = 0..7;
    diagnosis_rec = record
                      option_no : options;
                      no_of_questions : integer;
                      correct_answers : integer
                    end;
    diagnostic_rec = array[1..6] of diagnosis_rec;
    x_coordinates = 1..80;
    y_coordinates = 1..24;


var
    option : options;
    chapter : text;
    header : string;
    diagnosis : diagnosis_rec;
    diagnostics : diagnostic_rec;
    x : x_coordinates;
    y : y_coordinates;



{procedures}

procedure clear_screen;

{This routine clears the screen and leaves the cursor in the top left
hand corner.}

begin {clear_screen}
    write(chr(12))      {12 is the ASCII code number for clear screen}
end;  {clear_screen}


procedure gotoxy(x : x_coordinates;y : y_coordinates);

{This routine takes the cursor to the 'home' position (top left)
and then it uses  the coordinates x and y and  moves the cursor to
the line y, character x position on the screen.  The maximum value
of x is 80 and y is 24.}
```

```
var
   count : integer;

begin {gotoxy}
   write(chr(27),'H');          {27 is the ASCII code number for ESC, the}
   for count := 1 to y do       {effect of 'ESC H' is the same as 'HOME'.}
      write(chr(27),'B');       {This represents 'ESC B' which is cursor down}
   for count := 1 to x do
      write(chr(27),'C')        {This represents 'ESC C' which is cursor right}
end;  {gotoxy}


procedure draw_line(y : y_coordinates);

{Using the alternative character set, this routine draws a straight line
at the specified line number.}

var
   count : integer;

begin
   write(chr(14));              {switches the alternative character set on.}
   gotoxy(5,y);
   for count := 1 to 70 do      {the ' ' is a straight line character in    }
      write(' ');               {the alt chr set.                           }
   write(chr(15))               {switches the normal character set on.      }
end;


procedure split_screen_1;

{The screen is split at lines 2,4 and 21 in this routine.}

begin {split_screen_1}
   draw_line(2);
   draw_line(4);
   draw_line(21)
end;  {split_screen_1}


procedure split_screen_2;

{The screen is split at lines 3 and 21 in this routine.}

begin {split_screen_2}
   draw_line(3);
   draw_line(21)
end;  {split_screen_2}


procedure continue;

{When this routine is evoked the message in quotes is displayed at line 22.}

var
   ch : char;

begin {continue}
   gotoxy(5,22);write('When you are ready, press any key to continue. ');
   read(ch);
   gotoxy(1,22);writeln(' ':80)
end;  {continue}


procedure display_menu;
```

```
{After the screen has been cleared and defined by the lines drawn by
 split_screen_1, this  routine  displays  the  chapter titles in this
 project 'syllabus'.}

begin {display_menu}
    clear_screen;
    split_screen_1;
    gotoxy(30,3);writeln('Master Screen');
    gotoxy(1,10);
    writeln('1.      Force of Gravity, Weight and Friction.':59);
    writeln('2.      Speed, Velocity and Acceleration.      ':60);
    writeln('3.      Newtons Laws of Motion.                ':60);
    writeln('4.      Work, Energy and Power.                ':60);
    writeln('5.      Machines.                              ':60);
    writeln('6.      Density and Relative Density.          ':60);
    writeln;
    writeln('7.      End this session.                      ':60);
end;  {display_menu}


function enumerated(ch : string):integer;

{This function takes the string ch and converts it into an integer.
 It does this by taking the left most character in ch and converting
 it to an integer value, then this is multiplied by 10 to the power
 of the number of positions the character is away from the right most
  character.  This process is repeated on all the characters.}

var
    num : integer;
    count : integer;


function power(x,y : integer):integer;

{This function returns the value x to the power of y.}

var
    count : integer;
    dum   : integer;

begin {power}
    dum := 1;
    for count := 1 to y do
        dum := dum * x;
    power := dum
end;  {power}


begin {enumerated}
    num := 0;
    for count := length(ch) downto 1 do
        num := (num + (ord(ch[count])-ord('0'))) * power(10,(count-1));
    enumerated := num;
end;  {enumerated}


procedure choose_option;

{This procedure is called after the menu has been displayed.  It asks
 the user for a number, coresponding to a chapter and validates it.}

var
    number : string;
    valid_option : boolean;
    valid_ans : integer;

begin  {choose_option}
    repeat
        gotoxy(1,22);write(' ':80);
```

```
        gotoxy(5,22);write('Type in the number of the chapter you require : ');
        readln(number);
        if enumerated(number) in [1..7] then valid_option := true
                                        else valid_option := false;
    until valid_option;
    valid_ans := enumerated(number);
    if valid_ans = 7 then option := finish else option := valid_ans
end;    {choose_option}


procedure open_file;

{When a valid option has been chosen, this procedure resets the variable
 chapter to the relevent CHAPTER.}

begin   {open_file}
    case option of
            1 : reset(chapter,'CHAPTER#01');
            2 : reset(chapter,'CHAPTER#02');
            3 : reset(chapter,'CHAPTER#03');
            4 : reset(chapter,'CHAPTER#04');
            5 : reset(chapter,'CHAPTER#05');
            6 : reset(chapter,'CHAPTER#06')
    end;
end;    {open_file}


procedure set_header;

{This routine assigns the variable header with the name of the
 current chapter.}

begin   {set_header}
    case option of
            1 : header := 'Force of Gravity, Weight and Friction';
            2 : header := 'Speed, Velocity and Acceleration';
            3 : header := 'Newtons Laws of Motion';
            4 : header := 'Work, Energy and Power';
            5 : header := 'Machines';
            6 : header := 'Density and Relative Density'
    end;
end;    {set_header}


procedure display_frame(flag : string);

{This routine reads the chapter text until the first occurance of
 the specified flag.  Then the screen is cleared and split into
 three parts.  In the top right hand corner the page number is
 written, then a frame (15 lines) of text are displayed and the user
 is asked to indicate when they want to continue.  Lines of text are
 displayed until the second occurance of the flag.}

var
    line : string;
    pnum, lnum : integer;

begin {display_frame}
    repeat readln(chapter,line) until (line = flag);
    readln(chapter,line);
    pnum := 0;
    while not(line = flag) do
    begin
        clear_screen;
        split_screen_1;
        pnum := pnum + 1;
        gotoxy(5,3);
        writeln(header);
        gotoxy(65,3);
        writeln('  Page ',pnum:2);
        gotoxy(1,5);writeln;
```

```
        lnum := 1;
        while not((lnum = 15) or (line = flag)) do
        begin
            writeln(' ':10,line);
            readln(chapter,line);
            lnum := lnum +1
        end;
        continue
    end;
end;  {display_frame}


procedure display_question(qnum : integer);

{This routine clears the screen and splits it into three parts.
The first part contains the number of the question being displayed,
the second part displays the question.  The routine reads the text
until the first occurance of the question flag and displays the
lines encountered before the second occurance of the flag.}

var
    line : string;

begin  {display_question}
    repeat readln(chapter,line) until (line = question_flag);
    readln(chapter,line);
    clear_screen;
    split_screen_1;
    gotoxy(5,3);
    writeln('Question ',qnum:1);
    gotoxy(1,5);writeln;
    while not(line = question_flag) do
    begin
        writeln(' ':10,line);
        readln(chapter,line);
    end;
end;  {display_question}


procedure present_text;

{The frames of text are displayed using the routine display_frame.}

var
    question_flag : string;

begin  {present_text}
    display_frame(text_flag)
end;  {present_text}


procedure present_questions;

{The record diagnosis is initialised in this routine and while the end
of chapter flag has not been encountered  it displays a question, reads
the answer from the chapter file asks for a valid answer from the user
and then marks this answer.}

var
    question_flag, line : string;
    correct_ans, ans : char;
    qnum : integer;


procedure good_response;

{This routine is called if the user has answered a question correctly.
It tells the users that he/she is correct and then skips over the
section of text that serves as reinforcement.  It also updates the
diagnosis record.}
```

```
var
    line : string;

begin  {good_response}
    gotoxy(25,19);writeln(' ':80);
    gotoxy(25,19);writeln('Yes ',ans,' is the correct answer.');
    repeat readln(chapter,line) until (line = reinforcement_flag);
    repeat readln(chapter,line) until (line = reinforcement_flag);
    with diagnosis do
    begin
        no_of_questions := no_of_questions + 1;
        correct_answers := correct_answers + 1
    end;
    continue
end;   {good_response}


procedure bad_response;

{This routine is called if the users has answered a question incorrectly.
 He/she is told what the answer should have been and the told that
 reinforcement text will be displayed explaining the answer.  After this
 text has been displayed the diagnosis record is updated.}

var
    ch   : char;
    flag : string;

begin  {bad_response}
    write(chr(7));  {This produces a noise from the terminal.}
    gotoxy(25,19);writeln(' ':80);
    gotoxy(25,19);writeln(ans,' is incorrect, the answer is ',correct_ans);
    gotoxy(5,22);write('Press any key for an explanation of this answer.');
    read(ch);
    header := 'Reinforcement text';
    display_frame(reinforcement_flag);
    with diagnosis do
        no_of_questions := no_of_questions + 1
end;   {bad_response}


begin  {present_questions}
    with diagnosis do
    begin
        option_no := option;
        no_of_questions := 0;
        correct_answers := 0
    end;
    qnum := 0;
    while not (line = end_of_chapter_flag) do
    begin
        qnum := qnum + 1;
        display_question(qnum);
        repeat readln(chapter,line) until (line = answer_flag);
        readln(chapter,correct_ans);
        repeat readln(chapter,line) until (line = answer_flag);
        repeat
            gotoxy(25,19);writeln(' ':80);
            gotoxy(25,19);write('Type in either A B or C : ');
            readln(ans);
            {The next line changes lower case letters to uppercase}
            if (ans >= 'a') then ans := chr(ord(ans)-32);
        until ((ans = 'A') or (ans = 'B') or (ans = 'C'));
        if (ans = correct_ans) then good_response
                               else bad_response;
        readln(chapter,line)
    end;
end;   {present_questions}


procedure update_stats;
```

```
{The record diagnosis is added to the list of diagnostics.}

begin  {update_stats}
    diagnostics[option] := diagnosis
end;   {update_stats}


procedure analyse_session;

{When the user has chosen the option to finish this routine is executed
to produce the marks.  If a user has completed a chapter then the
percentage gained in the chapter is calculated and displayed with the
chapter name.  If a chapter has not been completed then a not attempted
comment is displayed with the chapter name.  If no chapters were
attempted then the routine finishes otherwise it displays an overall
percentage before completion.}

var
    count : integer;
    percentage,total_correct,total_questions : integer;
    none_attempted : boolean;

begin  {analyse_session}
    clear_screen;
    split_screen 1;
    gotoxy(28,3);writeln('D I A G N O S T I C S');
    total_correct := 0;
    total_questions := 0;
    none_attempted := true;
    for count := 1 to 6 do
    begin
        option := count;
        set_header;
        if not(diagnostics[count].option_no = null)
            then with diagnostics[count] do
            begin
                percentage := round((correct_answers/no_of_questions)*100);
                writeln;write(' ':5);
                writeln('In the chapter "',header,'" you gained ',percentage:1,'
                total_correct := total_correct + correct_answers;
                total_questions := total_questions + no_of_questions;
                none_attempted := false
            end
            else
            begin
                writeln;write(' ':5);
                writeln('The chapter "',header,'" was not attempted.')
            end;
     end;
    gotoxy(27,20);
    if not(none_attempted) then
    begin
        percentage := round((total_correct/total_questions)*100);
        writeln('Your overall mark is ',percentage:1,' % ')
    end;
    gotoxy(1,22);
    writeln
end;   {analyse_session}


procedure initialise;

{The diagnostics records are initialised in this routine.}

var
    count : integer;

begin  {initialise}
    for count := 1 to 6 do
        with diagnostics[count] do
        begin
            option_no := null;
            no_of_questions := null;
```

```
            correct_answers := null
        end;
end;    {initialise}


procedure titles;

{The title of the program and name of its author are displayed using
this routine.}

begin  {titles}
    clear_screen;
    split_screen_2;
    gotoxy(30,10);writeln('M E C H S ');
    gotoxy(30,11);writeln('--------- ');
    gotoxy(20,13);writeln('A Program to teach ''O'' level mechanics.');
    gotoxy(45,14);writeln('by S J Jabbar.');
    continue;
end;    {titles}


procedure introduction;

{The user is asked whether he/she would like an introduction to the
 program.  If they do then the file containing this is displayed
 on split_screen_2.}

var
    flag, ans : string;

begin  {introduction}
    clear_screen;
    split_screen_2;
    gotoxy(20,11);writeln(' ':10);
    gotoxy(20,11);write('Would you like an introduction ? ');
    readln(ans);
    if ((ans = 'YES') or (ans = 'Y') or (ans = 'yes') or (ans = 'y')) then
    begin
        reset(chapter,'INTRO.TEX');
        header := 'Introduction.';
        display_frame(introduction_flag)
    end;
end;    {introduction}


begin  {main body}
    titles;
    introduction;
    display_menu;
    initialise;
    choose_option;
    while not(option = finish) do
    begin
        open_file;
        set_header;
        present_text;
        present_questions;
        update_stats;
        display_menu;
        choose_option;
    end;
    analyse_session
end.   {main body}
```

INTRODUCTION : INTRO.TEX

{I}

This program will attempt to teach you some mechanics. Mechanics is a branch of physics which is concerned with the relation of matter to energy.

You will be presented with six chapters any of which you may attempt. These chapters have some pages of text which explain the subject matter.

After you have read the chapter you will be asked to try and answer some questions on the chapter. Reinforcement text will be supplied should it be deemed necessary.

When you have finished, an analysis of the session is displayed showing how well you answered the questions from the chapters attempted. You will also be given an overall mark.
{I}

FORCE

Force is that which changes a body's state of rest or of uniform motion in a straight line.

In general terms this means that when a body is acted upon by a force it will begin to move. If the body is already moving a force may alter its speed or alter its direction of motion or bring it to rest.

The unit of measurement for force is called the newton (N) and is defined as follows :

One newton is the force required to give a mass of one kilogram an acceleration of one metre per second every second.

In symbols it can be written as:

given, $F = ma$

if $m = 1$ kg (mass)
and $a = 1$ m/s^2 (acceleration)

then $F = 1$ N (Force)

GRAVITATIONAL FORCE

There is a force that acts on us constantly and this is the force that pulls us towards the earth. This is called gravitational force. Sir Isaac Newton concluded that gravitational force exists between all bodies.

Newtons law of universal gravitation states that any two particles of matter attract one another with a force which is proportional to the product of their masses and inversely proportional to the square of their distance apart.

So two stones are not only attracted towards the earth but also to each other.

Gravitational attraction is also responsible for holding the moon in its orbit about the earth and also the earth and its fellow planets in their orbits round the sun. This particular force, which keeps a body moving in a circle, is a force directed towards the centre. It is called centripetal force.

WEIGHT

Weight is the name of the force due to gravity and experiment has shown that on average on the earth, the force of gravity on a mass of one kilogram is 9.8 N.

So the weight of one kilogram is 9.8 N.

WEIGHTLESSNESS

Weightlessness occurs when the earths gravitational pull is equal to the centripetal force required to maintain a particular speed and orbital radius. Since the difference between the gravitational force and the centripetal force is zero there would be no resultant force acting on the body. It would be weightless.

{T}

FRICTION

Friction is the name given to the force which opposes the relative sliding motion of two surfaces in contact with one another. Without friction trains would not be able to stop since the breaks work through friction.

There are two types of friction static and sliding.

STATIC FRICTION

if a gradually increasing force is applied to a body it will, at first, continue to remain at rest since an equally increasing but oppositely directed force of friction comes into actionat the under surface of the body. At any particular moment, we say that the pull and the opposing frictional force are in equilibrium.

reached when the body just begins to slip. At this point, the friction brought into play has reached its maximum value for the two surfaces concerned and this is called static friction.

SLIDING FRICTION

If the body is pulled along so that it slips at a steady speed we notice that the force required is rather less than the static frictional force. Sliding friction is therefore less than static.

COEFFICIENT OF FRICTION

In any body static and sliding friction are increased in proportion to the force, perpendicular to the surfaces, which is pressing them together.

The Ratios of the static and sliding friction to the force pressing the surfaces together are called the coefficients of static and sliding friction respectively.

Coefficient of friction is denoted by the greek letter mu Thus:

$$\mu = \frac{F}{R}$$

where F = force of friction
and R = force of pressing surfaces together

{T}
{-Q}

A car of mass 1000 kg moves with an acceleration of 2 m/s^2. What is the force acting on it ?

A) 500 N

B) 998 N

C) 2000 N

{-Q}
{-A}
C
{A}
{-R}

The equation relating force, mass and acceleration is

{T}

F = ma

since m = 1000 kg
and a = 2 m/s^2

F = 1000 x 2 = 2000 N

{-R}
{Q}

The earth is not a perfect sphere but bulges at the equater. If an object was weighed at the pole and then weighed again but at the equater would its weight be:

A) Greater than before

B) Less than before

C) Still the same

{-Q}
{A}
B
{A}
{-R}

A force between two bodies, in this case the earth and the object, depends upon their masses (which is constant) and is inversely proportional to the square of their distance apart.

Since the distance to the centre of the earth is greater at the equater, the weight would be less.

{R}
{-Q}

Is it possible to walk on a perfectly smooth surface ?

A) Yes

B) No

C) It depends on mass

{Q}
{-A}
B
{A}
{R}

Walking is not possible if there is no friction between the ground and the soles of the shoe. A perfectly smooth surface is frictionless.

{R}
{-Q}

A force (R) pressing a body to a surface has a value of 50 N. The coefficient of friction (mu) is 2 and the body is accelerating (a) at 10 m/s^2.

What is the mass of the body ?

A) 1000 kg

B) 250 kg

C) 10 kg

{Q}
{A}
C
{-A}
{R}

This problem should be done in two parts since the equation linking the force (R) and the coefficient of friction (mu) does not give the mass. It does give the force of friction (F) and this can be combined with the acceleration (a) to give mass (m).

mu = F/R

therfore F = mu R
= 2 x 50
= 100 N

T}

t F = m a

erfore m = F/a
= 100 / 10

= 10 kg

R}
E}