



**TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS**

Project Title:

CES-D QUESTIONNAIRE

**A COURSE PROJECT SUBMITTED TO THE DEPARTMENT OF
ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE PRACTICAL
COURSE ON COMPUTER PROGRAMMING [CT 401]**

Submitted by:

Praphul Mishra (076/BAS/027)

Sandip Adhikari (076/BAS/035)

Pritam Shah (076/BAS/030)

Prithak Dahal (076/BAS/031)

Submitted to:

Department of Electronics and Computer Engineering

Pulchowk Campus

Institute of Engineering, Tribhuvan University

Lalitpur, Nepal

Falgun, 2077

Acknowledgement

First and foremost of all, our special gratitude goes to Mr. Santosh Giri for his guidelines and ideas for this project. We thank him for organizing this course and acknowledge his effort that encouraged us to take this challenging project and relate the programming. We also thank everyone who has helped us directly or indirectly in coming up with the idea of this project like fellows; who provided us with their valuable comments and suggestions regarding the project including the authors whose books we used as a reference, to gain the knowledge and information required for the accomplishment of this project.

Finally, we would also like to offer our gratitude to the representatives of Positive Psychological Center at University of Pennsylvania(Pennsylvania, USA) and all our teachers whose lectures and ideas were the basis for our project research and appreciate the support rendered by the Department of Electronics & Computer Engineering (DOECE), IOE. Without all your support and guidance the completion of this project would have been impossible.

ABSTRACT

The programming languages have become important to develop computer programs. It is essential to write efficient programs in order to solve all simple to complex problems. An efficient and powerful high-level language like C helps to solve large and complex problems in reasonable time. Learning C as a programming language is the first step for those who want to enter the profession of computer programming. As the programming language C gives standard construct, it is easier to learn any other language if one has a clear concept of C.

The main objective of this project is to develop a 20 items questionnaire through which depression symptoms in a person can be measure (not precisely) based on the responses to those items. This program can store the details of any testee including the CES-D index and later, delete the stored information as per the testee's requirement. Overall, this system provides basic knowledge about the utility of the C-programming language.

TABLE OF CONTENTS

ACKNOWLEDGEMENT.....	i
ABSTRACT.....	ii
TABLE OF CONTENTS.....	iii
LISTS OF FIGURES AND ABBREVIATIONS.....	iv
Chapter One: INTRODUCTION.....	1
Chapter Two: REVIEW OF RELATED LITERATURES.....	2
Chapter Three: ALGORITHM DEVELOPMENT AND FLOWCHART.....	8
Chapter Four: IMPLEMENTATION AND CODING.....	21
Chapter Five: RESULTS AND DISCUSSIONS.....	24
Chapter Six: CONCLUSIONS.....	28
REFERENCES.....	29
APPENDIX A (SOURCE CODE).....	30

LIST OF FIGURES AND ABBREVATIONS

Figures:

3.5: Flowchart for int main().....	11
3.6: Flowchart for test().....	13
3.7: Flowchart for search().....	16
3.8: Flowchart for del().....	18
3.9: Flowchart for exit().....	20
Figure-4.1.1: Snapshot of home page.....	21
Figure-4.1.2: Snapshot of menu.....	21
Figure-4.1.2: Snapshot of termination of program.....	22
Figure-5.1: Snapshot of information to be entered.....	24
Figure-5.2: Snapshot of the instructions.....	24
Figure-5.3: Snapshot of the CES-D statements and responses.....	25
Figure-5.4: Snapshot of the obtained CES-D index.....	25
Figure-5.5: Snapshot of the stored information.....	26
Figure-5.6: Snapshot of the successful deletion of information.....	26

Abbreviation:

- CES-D – Center For Epidemiologic Studies – Depression

Chapter one: INTRODUCTION

This mini-project in C-programming is prepared with basic knowledge of high-level programming language. This project is used to measure the extent of depression symptoms experienced by a testee(over a past week) based on the 20 items measure(responses of the testee). We used various concepts of C-programming in order to complete this project. We also used various macro functions like gotoxy (), etc. to make this program user friendly to some extent. Data files are also used in order to store the CES-D record of a testee.

1.1 Background and Problem statements

This project was done in accordance with the syllabus of Computer Programming [CT 452] which is included in first year second part of Bachelors of Aerospace Engineering.

It took us nearly one week to complete this project. After gaining the basic concepts from the lecture classes we started to write pseudo code primarily and then we finally changed the pseudo code into a program using the C-programming language. Although, the basic concepts were acquired in the lecture class, we had to do some research on our own to meet the requirements.

1.2. Objectives:-

- To gain knowledge about the C programming language.
- To become familiar with the functions, statements, array and other many functions of C and use it for our program.
- To understand how the C-programming works and how can we use it have a desired outputs.
- To understand the use of flowchart and algorithm in program.
- To build up the team co-ordination ability and boost up the team work confidence.

1.3. Limitations:- .

- The program is less user friendly.
- This program will not run in 16bit Operating System.

Chapter Two: REVIEW OF RELATED LITERATURES

What is C?

C is a programming language developed at AT & T's Bell Laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie. In the late seventies C began to replace the more familiar languages of that time like PL/I, ALGOL, etc. No one pushed C. It wasn't made the 'official' Bell Labs language. Thus, without any advertisement, C's reputation spread and its pool of users grew. Ritchie seems to have been rather surprised that so many programmers preferred C to older languages like FORTRAN or PL/I, or the newer ones like Pascal and APL. But that's what happened. Possibly why C seems so popular is because it is reliable, simple and easy to use.

1. Data Input and Output

Generally, a computer program takes input from keyboard (standard input device) or any other devices (e.g. data files). The program then processes the input and sends the result to output devices (i.e. computer screen or monitor or data file).

Formatted I/O:

Syntax for formatted I/O are:

```
scanf ("control string", &arg1, &arg2, . . . . , &argn);  
printf ("control string", arg1, arg2, . . . . , argn);
```

Unformatted I/O:

Syntax for unformatted I/O are:

```
character_variable = getchar();  
putchar (character_variable);  
character_variable = getch();  
character_variable = getche();  
putch (character_variable);  
gets (string_variable);  
puts (string_variable);
```

2. Control Statements

The statements which alter flow of execution of a program are known as control statements. In the absence of control statements, the instructions or statements are executed in same order in which they appear in source program. Such type of construct is known as sequential construct. There are mainly two types of control statements: Decision Making & Loop statements.

2.1 Decision Making Statements

The decision-making statements test a condition and allow to execute some statements on the basis of result of the test (i.e. either true or false). Since they have capability to decide whether specified statements have to be executed or not, they are

called decision making statements.

if Statement

An if statement is used to control the flow of execution of statements.

Syntax:

```
if (test_expression)
{
statement-block;
}
```

Nested if Statement

When one if statement is written within body of another if statement, it is called Nested if Statement.

Syntax:

```
if (expression)
{
if (expression)
{
body;
}
}
```

if..else Statement

The if..else statement is an extension of the simple if statement. It is used when there are two possible options.

Syntax:

```
if (test_expression)
{
true-block statements;
}
else (test_expression)
{
false-block statements;
}
```

if...else if Statement

An if..lse if statement is used when there are more than two possible action depending upon the outcome of test expression.

Syntax:

```
if (condition1)
{
Statement-1;
}
else if (condition2)
{
Statement-2;
}
```



```
else if (condition)
{
Statement-n;
}
```

```
else
{
default-statement;
}
```

2.2 Loop construct

Loop may be defined as the block of statements which are repeatedly executed for a certain number of times or until a particular condition is satisfied.

for Loop

The for loop allows to execute block of statements for a number of times.

Syntax:

```
for (counter_initialization; test_condition; increment or decrement)
{
statements or body of loop;
}
```

```
statements or body of loop;
}
```

while Loop

In this loop, the test condition is evaluated first and if the condition is true, the body of loop is executed.

Syntax:

```
while(test_condition)
{
body of loop;
}
```

do-while Loop

do-while loop executes a body (i.e. block of statements) first without checking any condition and then checks a test_condition is checked to determine whether the body of loop is to be executed for the next time or not.

Syntax:

```
do
{
statements or body of loop;
}while(test_condition);
```

break statement:

The break statement terminates the execution of the loop and the control is transferred

to the statement immediately following the loop.

Syntax:

```
break;
```

switch Statement

When there are a number of options available and one of them is to be selected on the basis of some criteria, switch statement is used.

Syntax:

```
switch (variable or expression)  
{  
  case caseConstant1:  
    statements;  
    break;  
  case caseConstant2:  
    statements;  
    break;  
  .  
  .  
  default:  
    statements;  
}
```

3. Function

A function is defined as self-contained block of statements that performs a particular specified job in a program. Actually, this is a logical unit composed of a number of statements grouped into a single unit. C functions can be classified into two categories: user-defined and library functions.

3.1 Library Functions (Built-in Functions)

These are the functions which are already written, compiled and placed in C library and they are not required to be written by programmer again. For example: *printf()*, *scanf()*, *sqrt()*, *getch()*, etc.

3.2 User-defined Functions

These are the functions which are defined by user at the time of writing a program. The user has choice to choose its name, return type and argument and their types.

4. Array and Strings

An array is a group of related data items that share a common name. In other words, an array is a data structure that store a number of data items as a single entity (object). The individual data items are called elements and all of them have same data types. For e.g. *int a[20];* is an integer array having maximum capacity to hold 20 array elements.

String is an array of character (i.e. characters are arranged one after another in

memory). In other words, a character-array is also known as a string. For e.g. *char ch[2][3];* is an string array.

5. Pointer

A pointer is a special type of a variable. It is special because it contains a memory address instead of values (i.e. data).

Syntax:

```
data_type *pointer_variable;
```

6. Structure

A structure is a collection of variables under a single name. The variables may be of different types, and each has a name which is used to select it from the structure.

Defining a structure:

Syntax:

```
struct structure_name
{
    data_type member_variable1;
    data_type member_variable2;
    ....
    ....
    data_type member_variablen;
}
```

Once structure_name is declared as a new data type, then variables of that type can be declared as:

```
struct structure_name structure_variable;
```

7. Data Files

A data file is a place on the disk where a group of related data is stored. The data file allows us to store information permanently and to access and alter that information whenever necessary. Programming language C has various library functions for creating and processing data files.

Opening and Closing a file

A program must open a file before to perform reading and writing operation on the file. Opening a file establishes a link between the program and the operating system. The buffer area is established as

```
FILE *ptr_variable;
```

A data file is opened using syntax:

```
ptr_variable=fopen (file_name, file-mode);
```

The file is closed using other library function fclose() as below:

```
fclose (ptr_variable);
```

File opening modes

The file opening mode specifies the way in which a file should be opened (i.e. for reading, writing or both, appending at the end of file, overwriting the file, etc).

w (write):

Open the file for writing purpose only.

r (read):

Open the file for reading purpose only.

a(append):

Open the file for appending or adding contents to it.

w+ (Write + Read):

Same as w except used for both reading and writing.

r+ (Read + Write):

The existing file is opened to beginning for both reading and writing.

a+ (append + read):

Same as a except both for reading and writing.

For example:

```
FILE *fp;
```

```
fp=fopen("student.txt","w");
```

```
/*The above statement opens file student.txt for writing purpose */
```

```
fp = fopen("student.txt", "r");
```

```
/*The above statement opens file student.txt for reading purpose. */
```




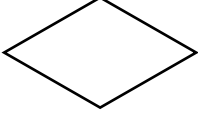




```
fp = fopen("student.txt", "a");
```

```
/*The above statement opens file student.txt for appending purpose. */
```

Chapter Three: ALGORITHM DEVELOPMENT AND FLOWCHART

An algorithm is a step-by-step description of the method to solve a problem. It is an effective procedure for solving a problem in a finite number of steps. Algorithm maintains sequences of computer instructions required to solve a problem in such a way that if the instructions are executed in the specified sequence, the desired result is obtained.

Flowcharts are usually drawn using standard symbols. Some standard symbols frequently required for flowcharting many computer programs are shown as below:

Symbols	Descriptions
	Arrow
	Start/Stop/End (Rounded Rectangle)
	Rectangle
	Decision
	Input/Output
	Connectors
	Function call
	For Loop

3.1 Algorithm for int main():

1. Start.
2. Display “CES-D QUESTIONNAIRE” and its contents.
3. Clear the screen.
4. Display the choices as 1, 2, 3, and 4.
5. Read the entered choice as num.
6. Is num = 1, 2, 3, or 4?
 - 6.1 If num 1, the module will goto test().
 - 6.2 Else if num 2, the module will goto search().
 - 6.3 Else if num 3, the module will goto del().
 - 6.4 Else if num 4, the module will goto exit().
 - 6.5 Else Display ”Invalid Choice” and goto 4.
7. Stop.

3.2 Algorithm for test():

1. Start.
2. Declare a[20] and file pointer *fp.
3. Read the details of the testee and store in structure array.
4. Display the instructions.
5. Clear the screen.
6. Display Statements.
7. Read the answer entered by the user for each statement and store in array a[20].
8. Assign 0 to b.score.
9. Declare i and assign 0 to i.
10. Check whether $i < 20$ or not. If yes continue, otherwise go to step 12.
11. Perform $b.score = b.score + a[i]$. Then, perform $i++$ and go to step 10.
12. Clear the screen.
13. Check whether $b.score \leq 16$ or not. If yes continue, otherwise go to step 15.
14. Display b.score indicating as No to mildly depressive symptomatology and go to 20.
15. Check whether $b.score > 16$ and ≤ 23 or not . If yes continue, otherwise go to step 17.
16. Display b.score indicating as moderately depressive symptomatology and go to 20.
17. Check whether $b.score > 23$ and ≤ 60 . If yes continue, otherwise go to 19.
18. Display b.score indicating as severely depressive symptomatology and go to 20.
19. Display “invalid score”.
20. Display a note clarifying about the obtained score.
21. Open the file “test.txt” in append mode.
22. Store the details of the testee in the file.

23. Close the file
24. Stop

3.3 Algorithm for search():

1. Start.
2. Declare san[20] and file pointer *fp1;
3. Clear the screen.
4. Read the entered name as san[20].
5. Open the file "test.txt" in reading mode.
6. Read the details from the file line by line.
 - 6.1 If details of a line coincide with the name, then go to step 7.
 - 6.2 Else go to step 8.
7. Display the matched details from the file and go to 10.
8. Display "This name doesn't exist".
9. Display "Do you want to search again(y/n)?".
10. Read the choice as ch.
 - 10.1 For ch == y, go to step 4.
 - 10.2 For ch == n, go to step 11.
 - 10.3 Else Display "Invalid choice" and go to step 9.
11. Close the file.
12. Stop.

3.4 Algorithm for del():

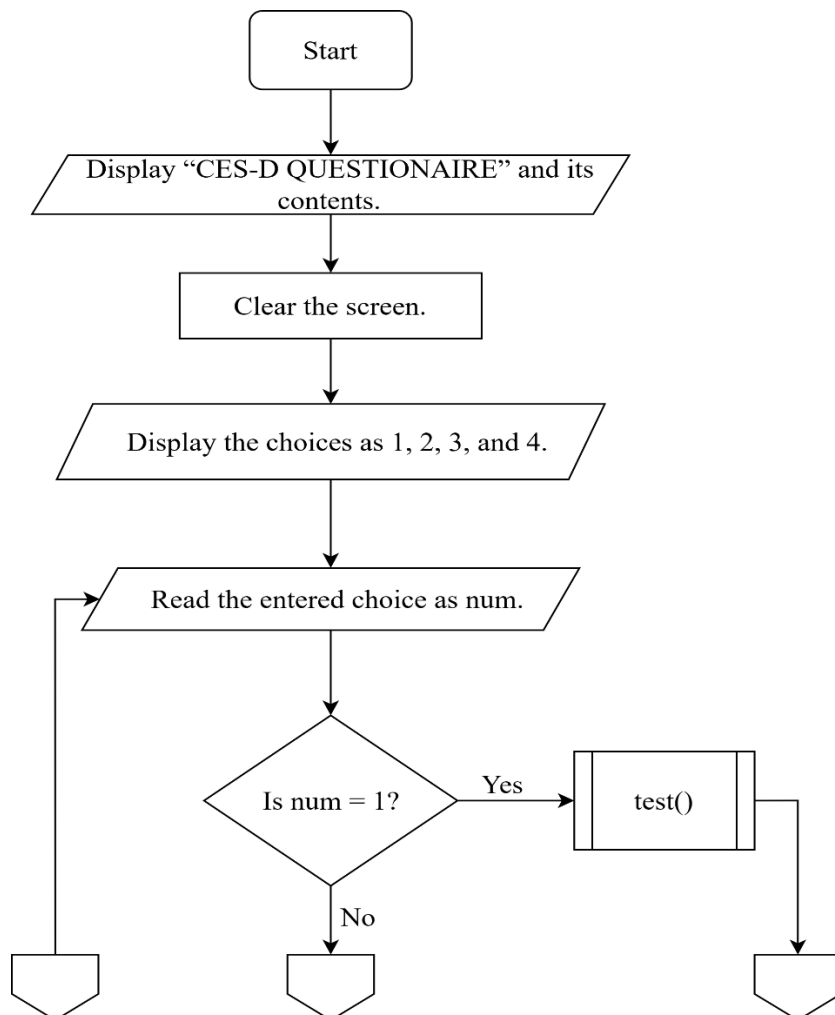
1. Start.
2. Declare sun[20], d=0, and file pointers *fp2 & *fp3.
3. Clear the screen.
4. Read the entered name as sun[20].
5. Open the files "temp.txt" and "test.txt" in writing and reading modes respectively.
6. Read the details from the file "test.txt".
 - 6.1 If some details match the name, then store the lines of unmatched details in the file "temp.txt", assign 1 to d and go to step 10.
 - 6.2 Else continue.
7. Display "This name doesn't exist".
8. Display "Do you want to try it again(y/n)?".
9. Read the choice as ch.
 - 9.1 For ch == y, go to step 4.
 - 9.2 For ch == n, go to step 11.
 - 9.3 Else Display "Invalid choice" and go to step 8.

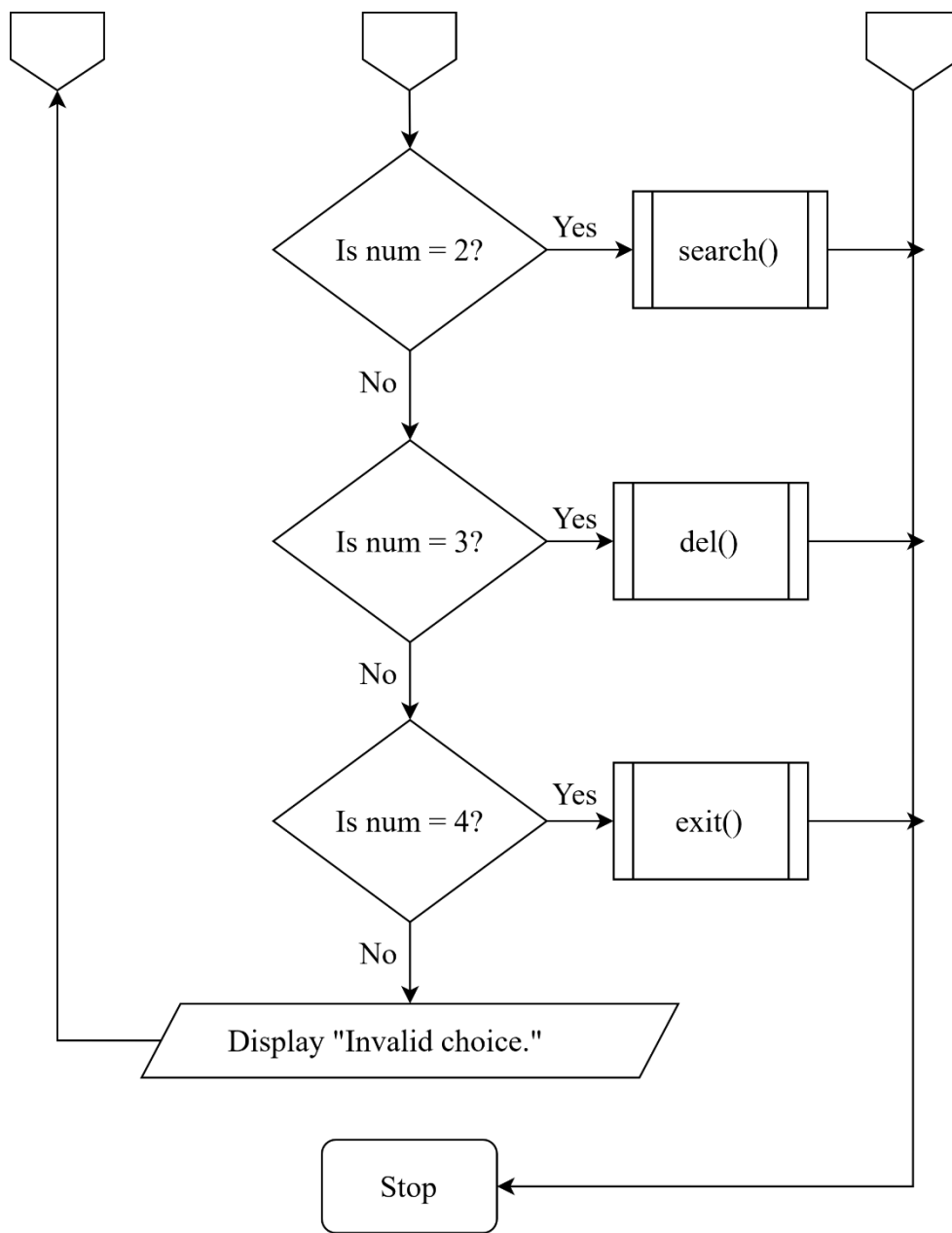
10. Close both files.
11. Delete the file “test.txt” and rename “temp.txt” as “test.txt”.
12. Check whether $d == 0$ or not. If yes continue, otherwise go to step 13.
13. Stop

3.5 Algorithm for exit():

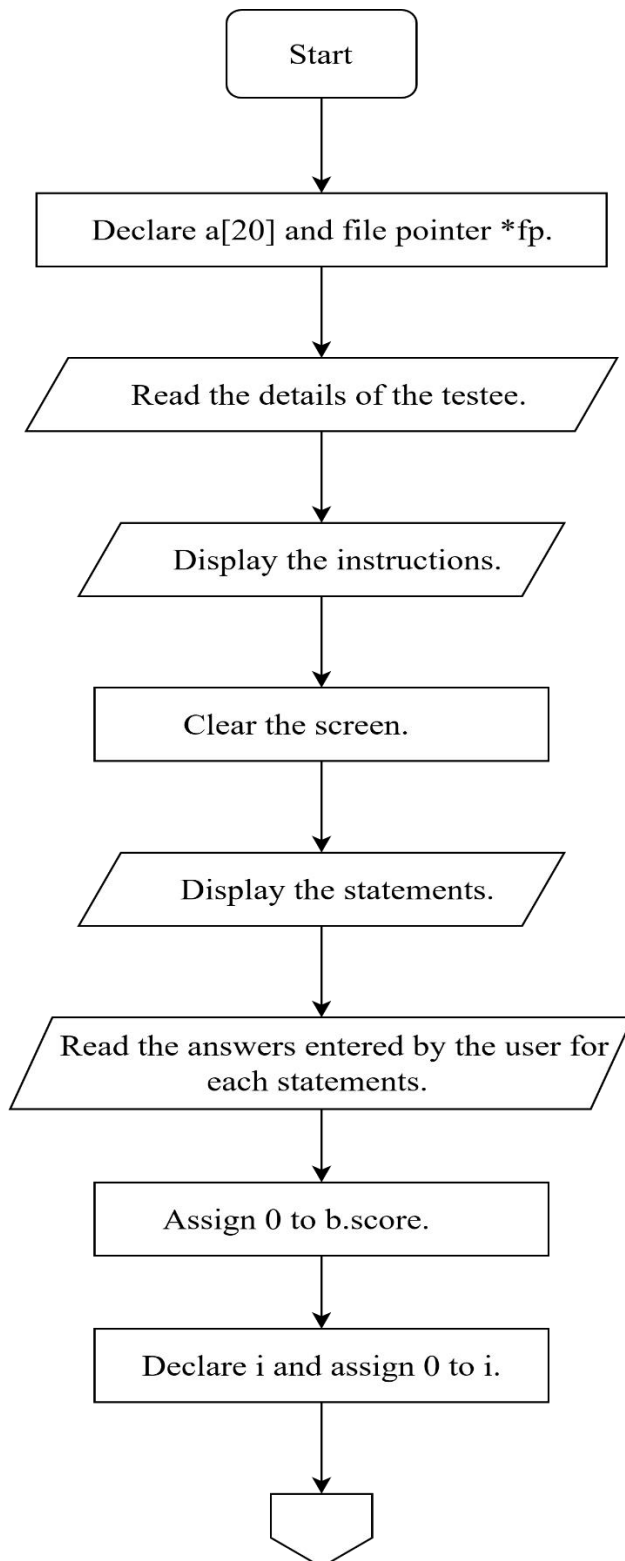
1. Start.
2. Display “Thank you for participating”.
3. Stop.

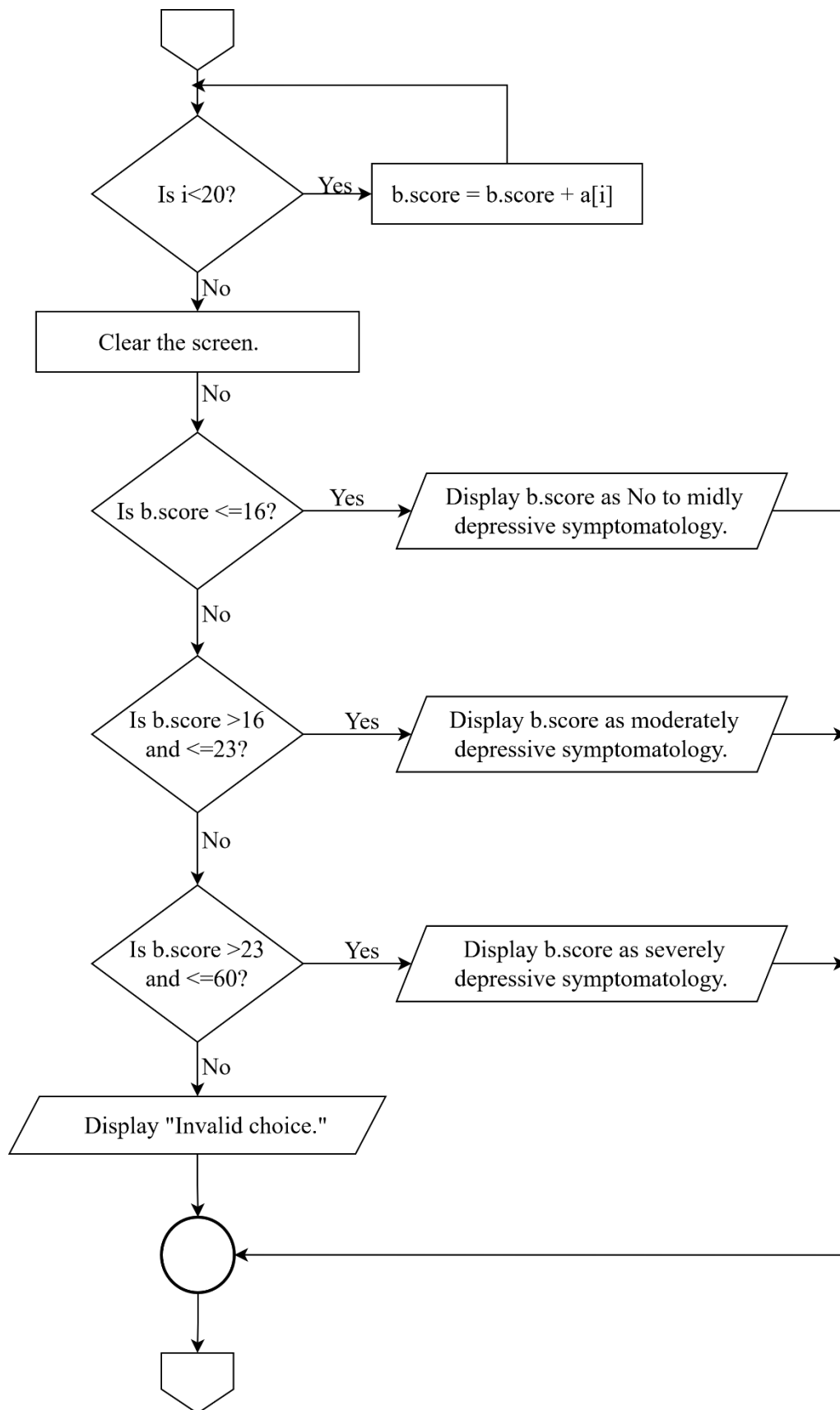
3.5 Flowchart for int main():

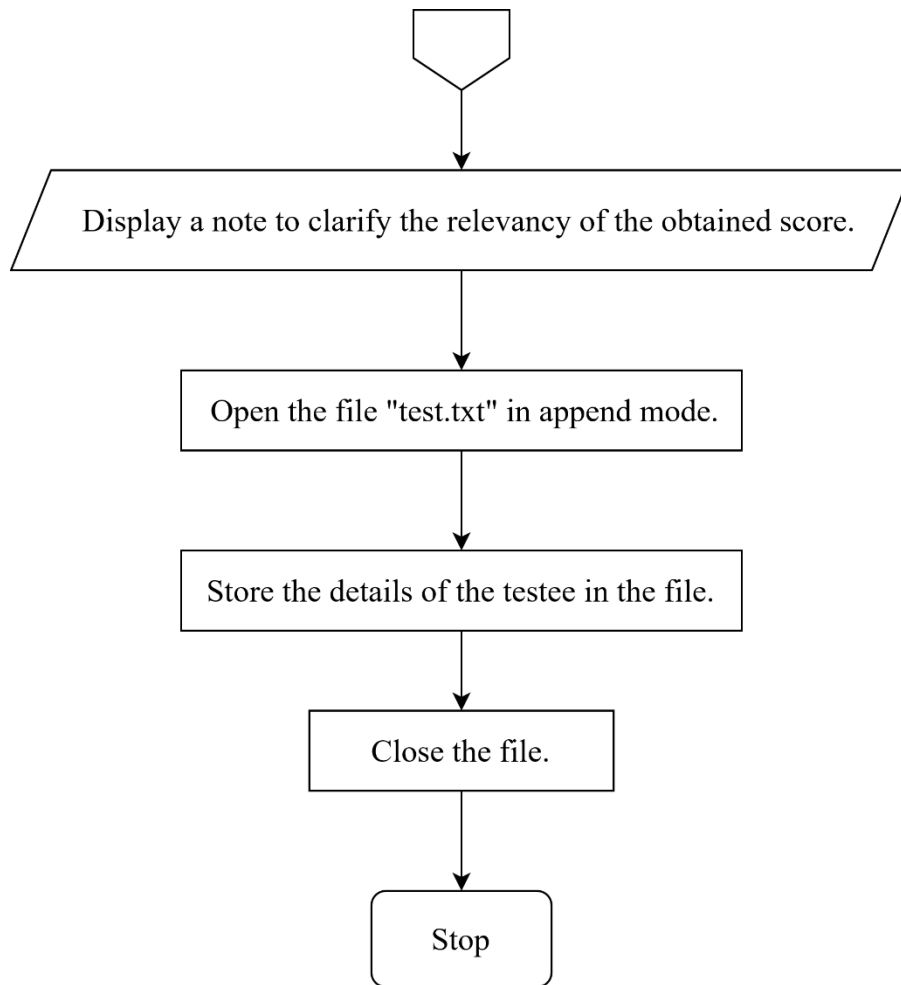




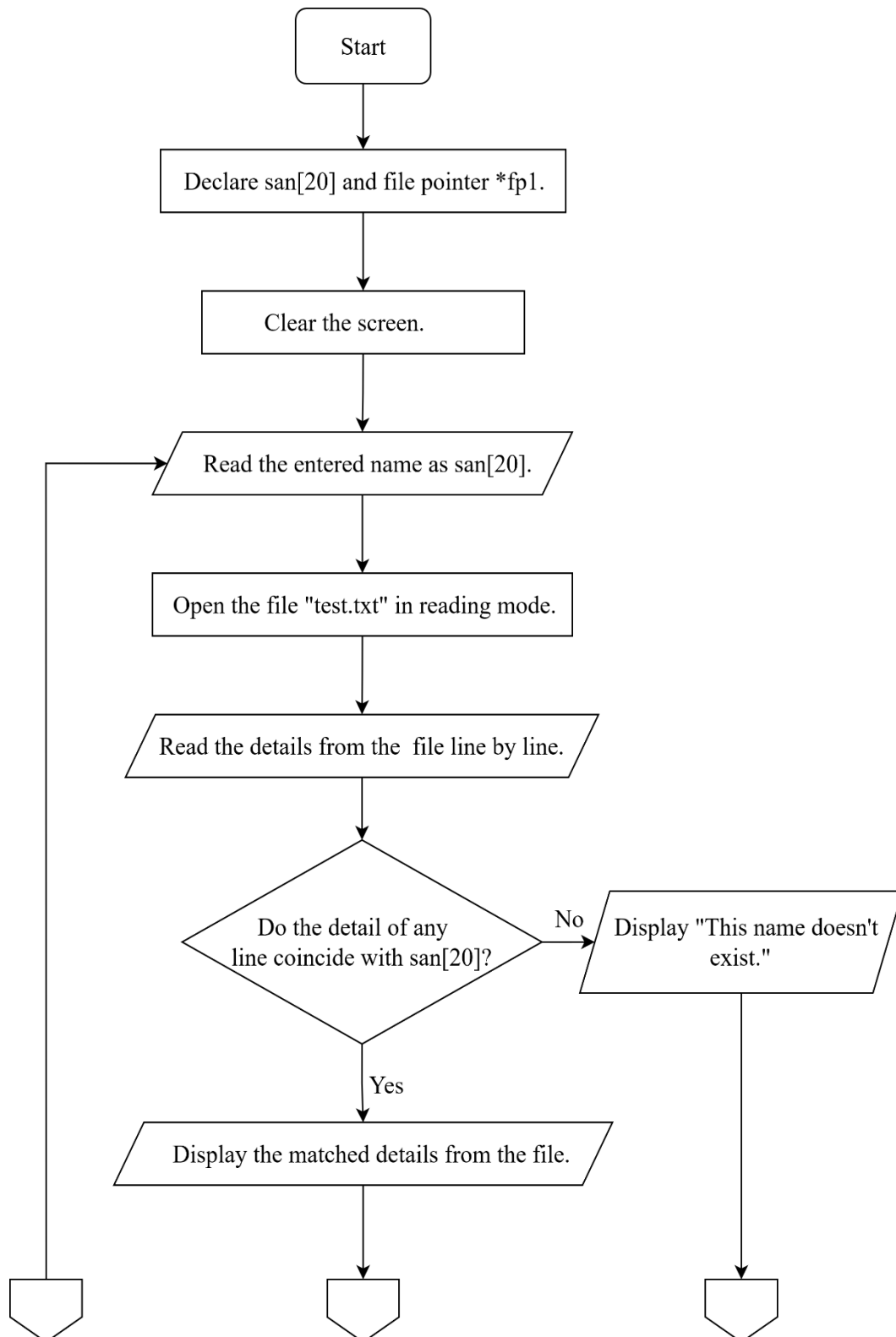
3.6 Flowchart for test():

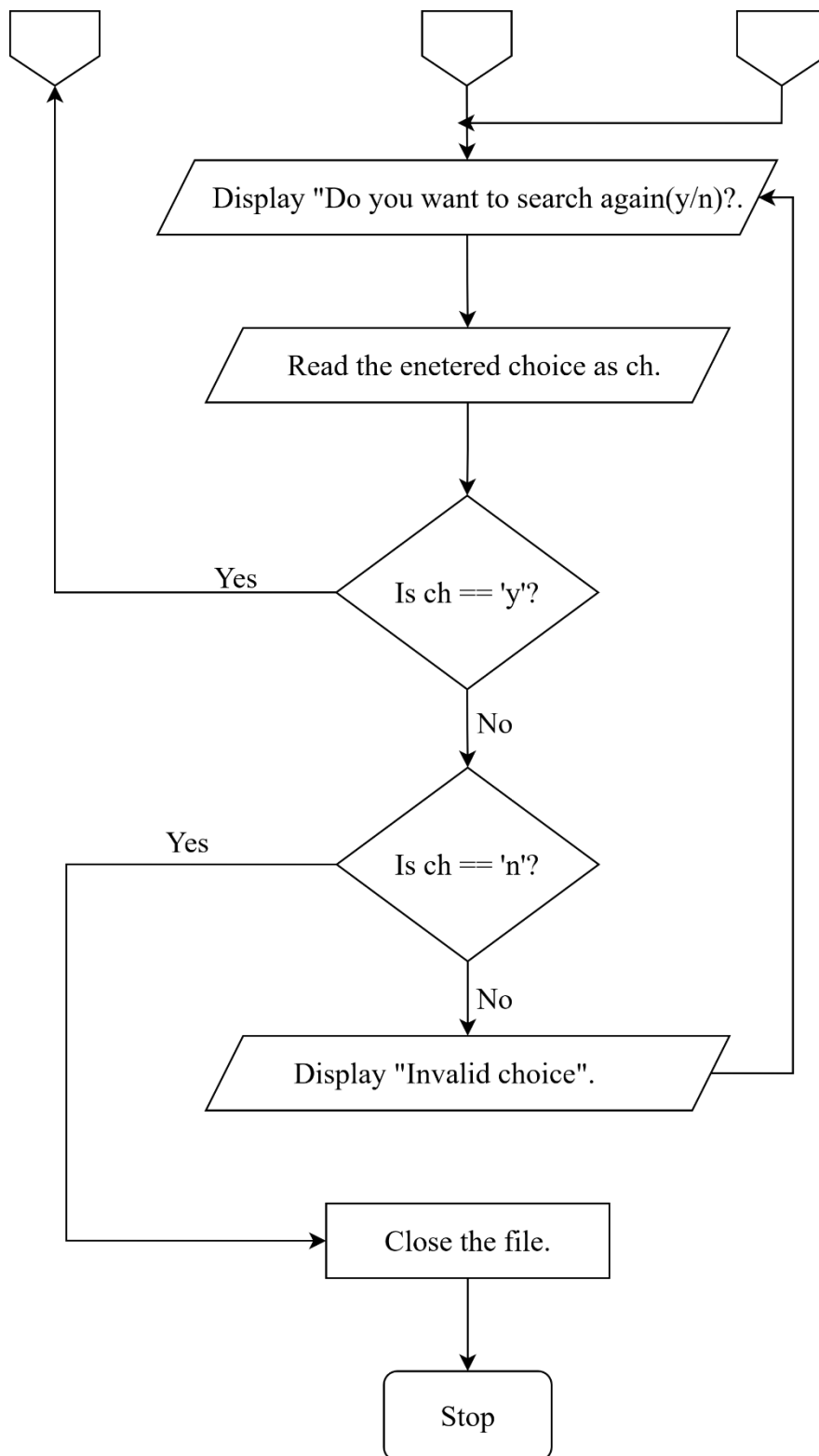




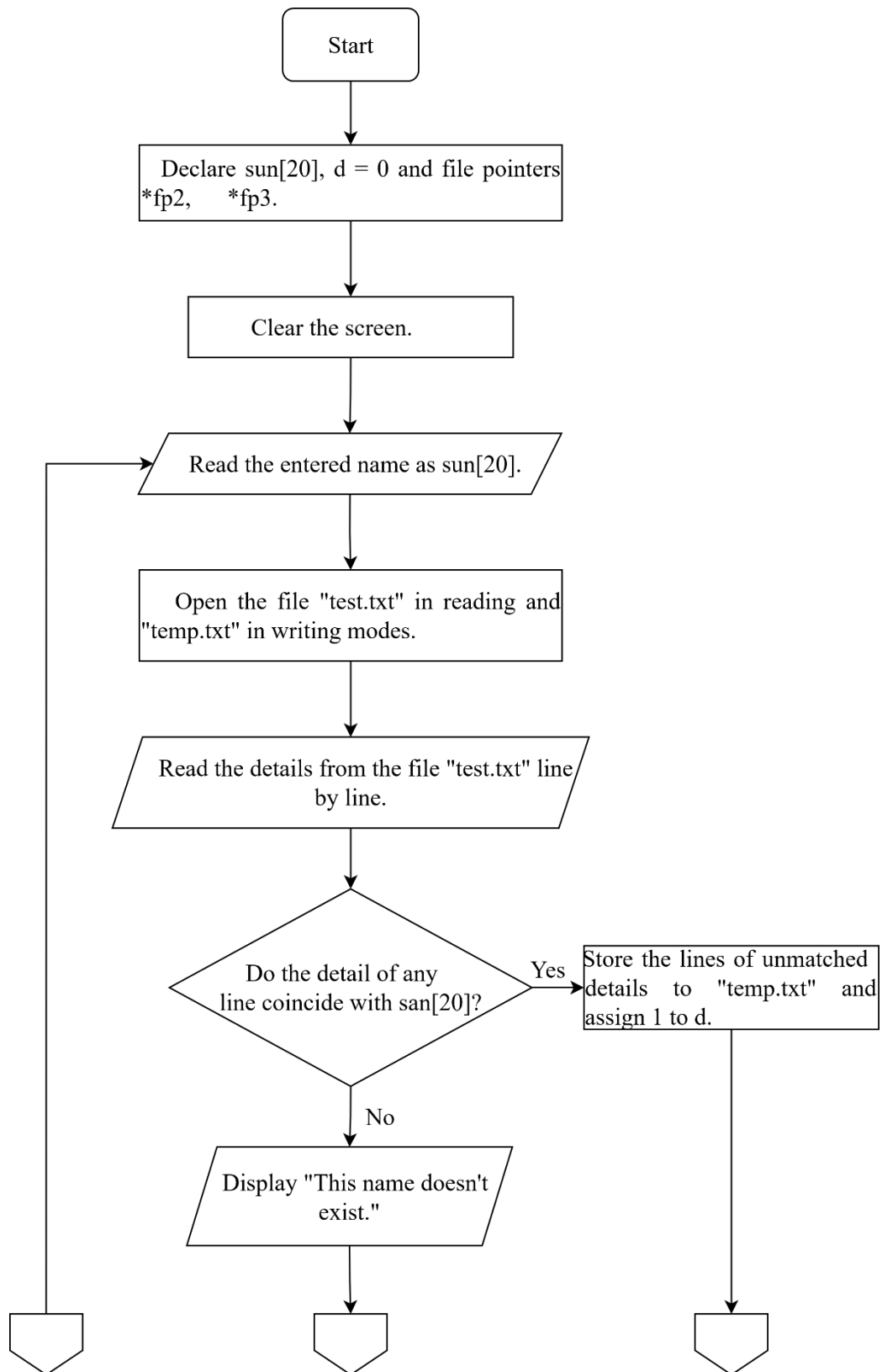


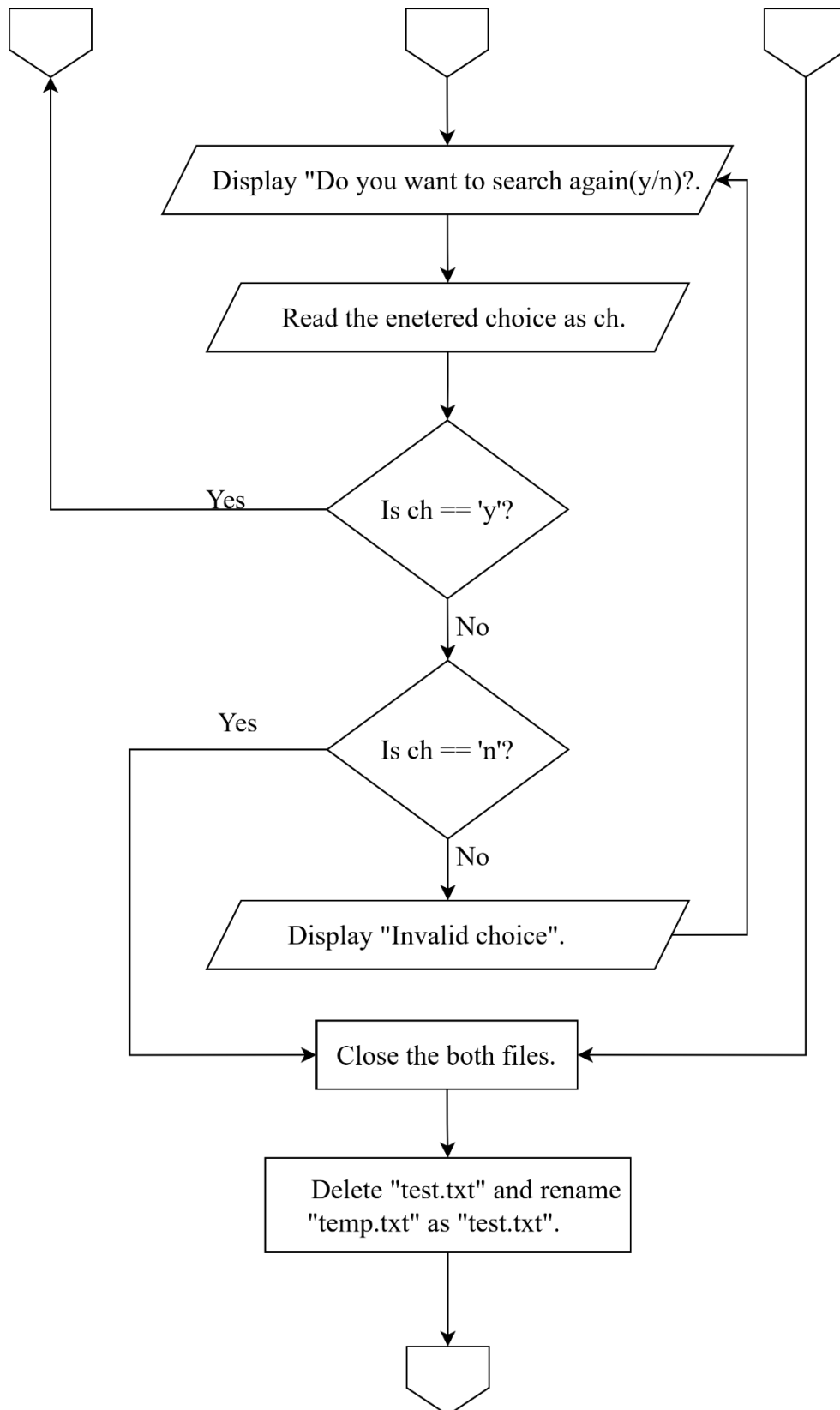
3.7 Flowchart for search():

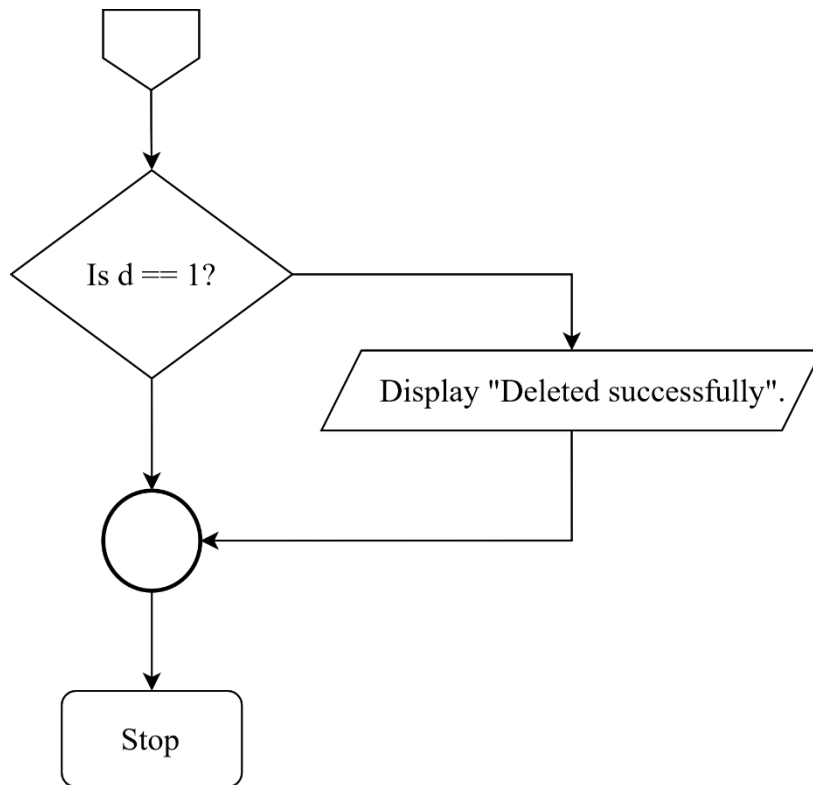




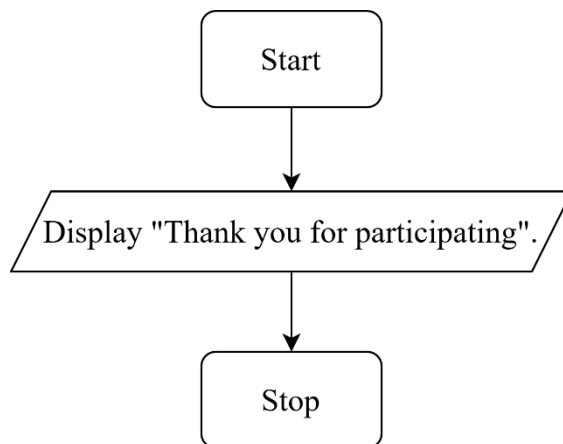
3.8 Flowchart for del():







3.9 Flowchart for exit():



Chapter FOUR: IMPLEMENTATION AND CODING

4.1 Implementation

The objective of the project has been achieved and the implementation of this program can be done as planned above. The program can be run easily on windows operating system with very less response time.

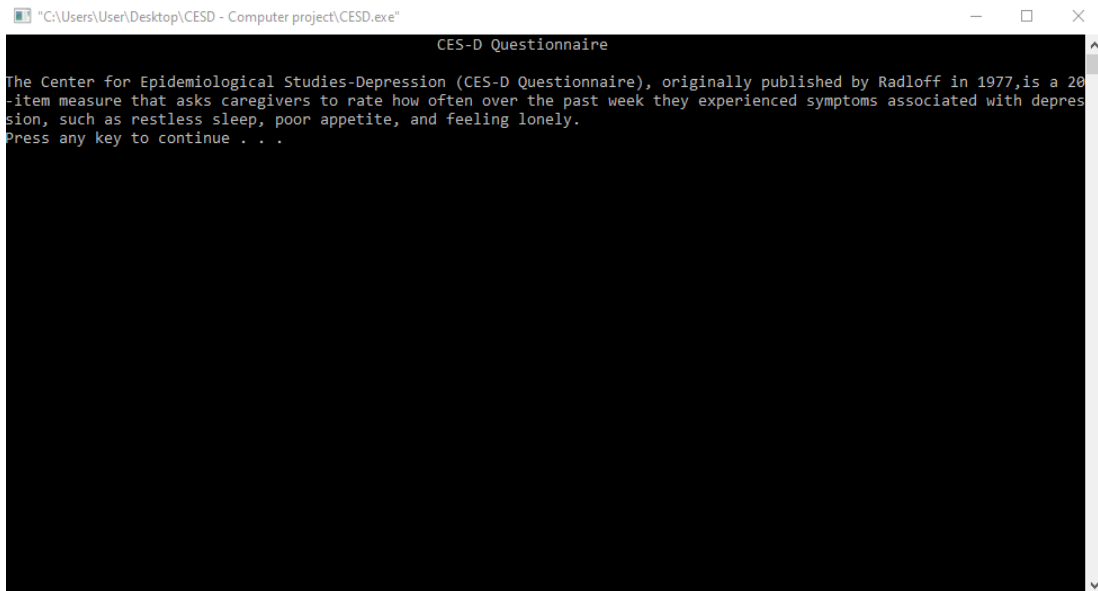


Figure – 4.1.1

Fig – 4.1.1 is the starting page of the program. It shows a brief description of the program and its applications.



Figure-4.1.2

Fig-4.1.2 is the menu of the program. Here, the user can choose an option (out of 4) from the menu by entering the corresponding number of the options.

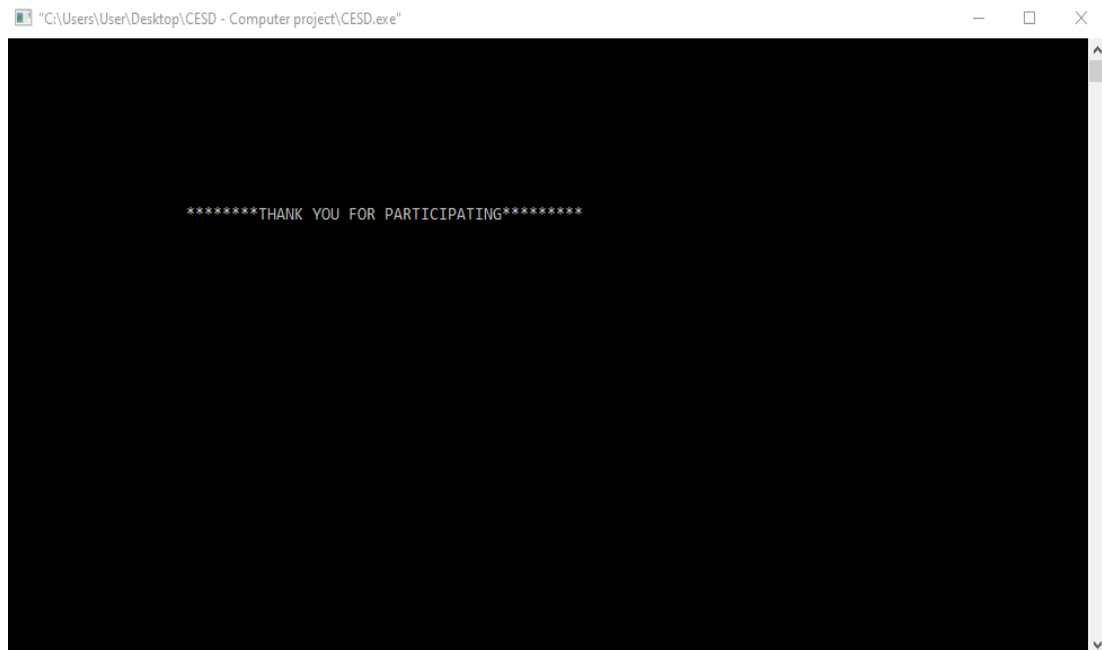


Figure-4.1.3

When the user enters the option 4, the program exits by displaying the content as shown in Fig-4.1.3

4.2 Coding of the project

In this project, C programming language is used and since C is a procedure-based programming language, the procedures involved in meeting the objective of this program are there in the source code. The flow of the program and the functions involved is explained below:

The functions used in this program are:

- `int main()`
- `void test()`
- `void search()`
- `void del()`
- `void exit()`
- `void gotoxy(int x, int y)`

The above functions work in the following way:

- **int main():** This is the main function of the program which is executed at

first. In this function, a switch construct is used in order to make the user to choose an option out of various. Here user needs to enter either 1, 2, 3, and 4 to choose “Take a test”, “Search for a testee’s CES-D index”, “Delete a Testee’s CES-D record”, or “Exit” respectively. User can choose any of the options. If the user enters an option other than those mentioned in the list then a message saying “Invalid choice.” is displayed. When the user enters a correct option, then the case constant is matched by the switch construct and the execution is done on the basis of the choice entered.

- **void test():** This function is of no return and no argument type. Here, some information about the testee is stored in the file named “test.txt”. The information includes Name, Email Id, and Contact no. along with the CES-D index which is obtained by adding the responses of the testee in each mentioned statements.
- **void search():** This function is used to find the details of a testee based on the name entered by the user. In this function, the name entered by the user is matched among the details stored in the file “test.txt” and the matched details are displayed.
- **void del():** This function is used to delete the CES-D index of a testee from the database based on the name entered by the user. Here, a temporary file “temp.txt” is created where the unmatched details from the file “test.txt” are stored. Then, “test.txt” is deleted and “temp.txt” is renamed as “test.txt”.
- **void exit():** This function is used for the termination of the program. Here, a grating message “Thank you for participating” is displayed.
- **void gotoxy():** This function takes the cursor to the coordinate as passed to this function. This function is basically used in order to manage the alignment in the program. The content in the printf () function specified after calling gotoxy () function is printed in the particular coordinate.

Chapter Five: RESULTS AND DISCUSSIONS

The results of the program can be better understood from the screenshots below:

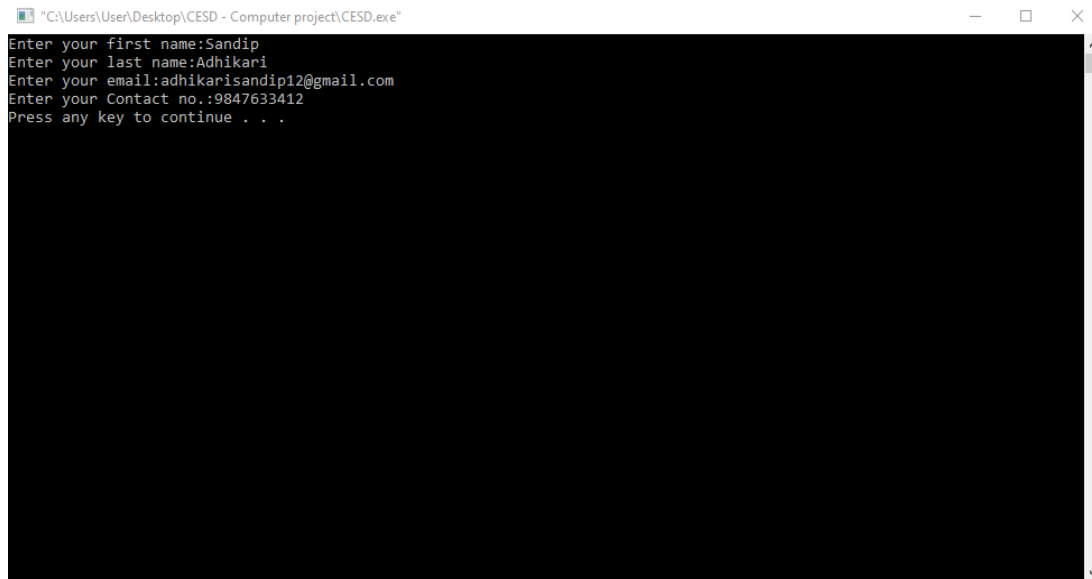


Figure-5.1

The window as shown in the fig-5.1 appears when the user enters choice 1 in the menu selecting the section “Take a test”. Fig-5.1 shows the information entered by the user or testee.

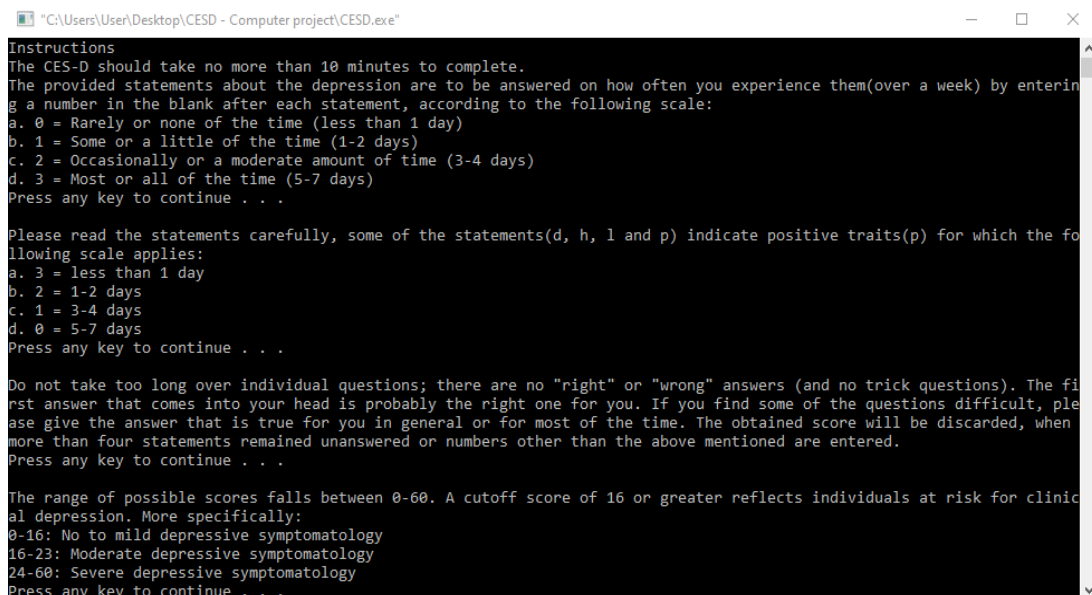


Figure-5.2

After the information is entered by the user, the window displays a set of instructions as shown in the fig-5.2; regarding the valid responses for CES-D questionnaire and scoring procedures of the CES-D index.

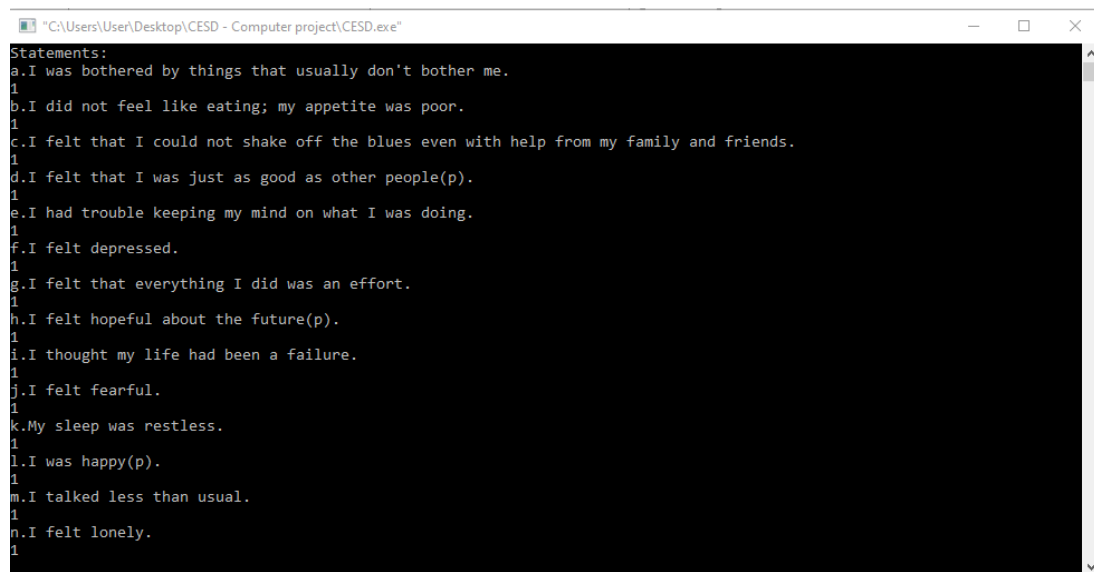


Figure-5.3

Fig-5.3 shows the statements of the CES-D Questionnaire and the corresponding responses given by the testee.

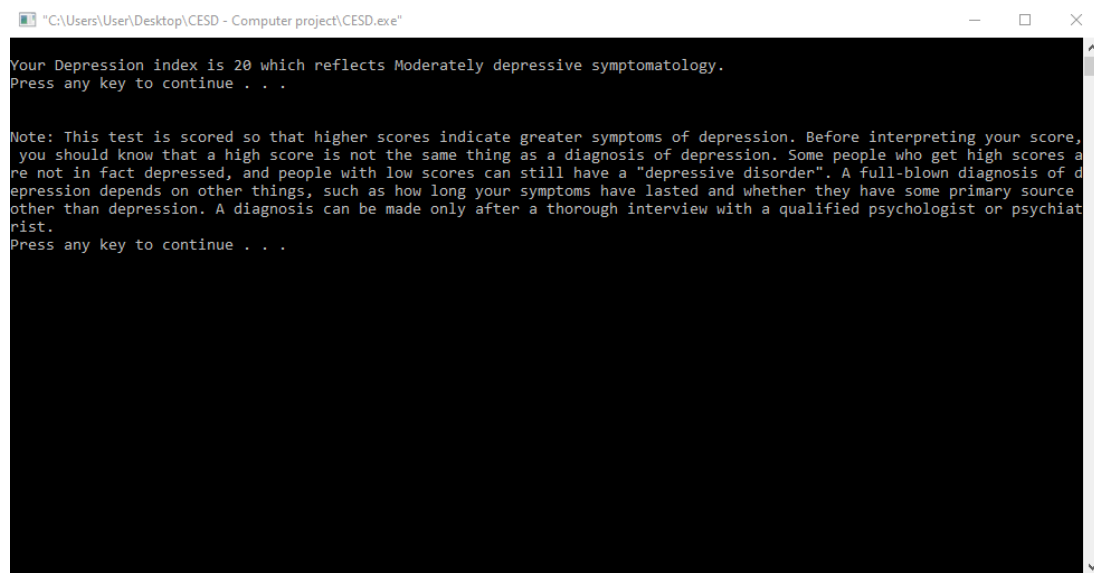
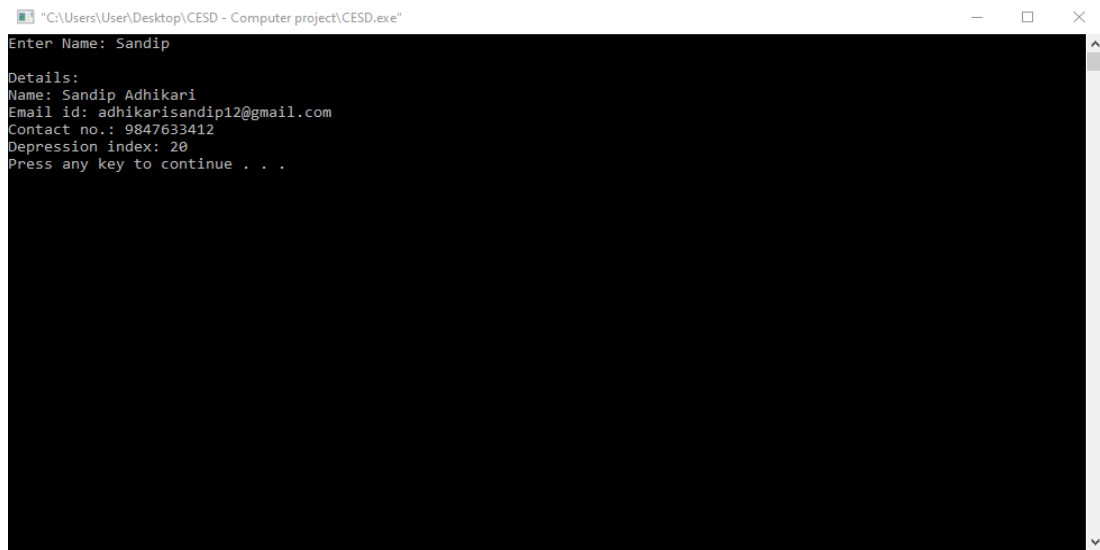


Figure-5.4

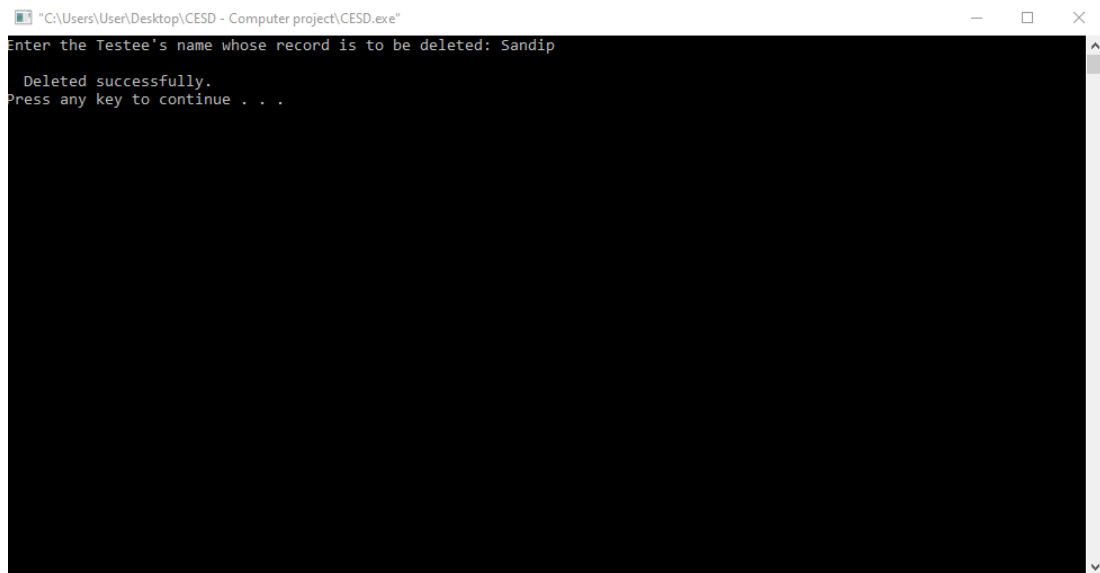
Fig-5.4 shows the depression index obtained by the testee and a note clarifying the relevancy of the obtained CES-D index.



```
"C:\Users\User\Desktop\CESD - Computer project\CESD.exe"
Enter Name: Sandip
Details:
Name: Sandip Adhikari
Email id: adhikarisandip12@gmail.com
Contact no.: 9847633412
Depression index: 20
Press any key to continue . . .
```

Figure-5.5

The window as shown in the fig-5.5 appears when the user enters choice 2 in the menu selecting the section “Search for a testee’s CES-D index”. Fig-5.5 shows the stored details of a testee based on the name entered by the user.



```
"C:\Users\User\Desktop\CESD - Computer project\CESD.exe"
Enter the Testee's name whose record is to be deleted: Sandip
Deleted successfully.
Press any key to continue . . .
```

Figure-5.6

The window as shown in the fig-5.6 appears when the user enters choice 3 in the menu selecting the section “Delete a testee’s CES-D record”. Here, the stored CES-D record of a testee is deleted based on the name entered by the user.

Chapter Six: CONCLUSIONS

After proper testing and executing the program, following conclusions can be made:

- This program is capable of measuring the extent of the depression symptoms (over a week) in a person based on the responses to 20 statements.
- The CES-D record of a testee can be stored and later, deleted as per the requirement of the user.
- This program also shows the proper utilization of concepts acquired in the lectures.
- This program is an example how simple programs can be developed using C programming.
- This program also helps in understanding the use of various functions in C programming.

REFERENCES

Kanetkar, Y. (2016). *Let Us C - 14th edition*. BPB Publications.

Ram Datta Bhatta, B. R. (2015). *A Textbook of C Programming*. Vidhyarthi Putsak Bhandar.

Er. Bikal Adhikari. (2019). *A Course on Computer Programming*. Heritage Publications

Authentic Happiness (Positive Psychology Center), University of Pennsylvania (Pennsylvania, USA)

APPENDIX A (Source Code)

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<windows.h>

struct information

{

    char namef[20],nameI[20], email[30], contact[20];

    int score;

};

struct information b;

char ch;

void test();

void search();

void del();

void Exit();

void gotoxy(int x, int y);

int main()

{

    printf("\t\t\t\t\tCES-D Questionnaire\t\t\t\n");

    printf("The Center for Epidemiological Studies-Depression (CES-D  
Questionnaire), originally published by Radloff in 1977,is a 20-item measure that asks  
caregivers to rate how often over the past week they experienced symptoms  
associated with depression, such as restless sleep, poor appetite, and feeling  
lonely.\n");

    system("pause");

    system("cls");

    char num;
```

```

start:

    gotoxy(20,7);
    printf("Menu");
    gotoxy(20,8);
    printf("1.Take a test\n");
    gotoxy(20,9);
    printf("2.Search for a Testee's CES-D index\n");
    gotoxy(20,10);
    printf("3.Delete a Testee's CES-D record\n");
    gotoxy(20,11);
    printf("4.Exit");
    gotoxy(20,12);
    scanf("%c",&num);
    switch(num)
    {
    case '1':
        system ("cls");
        test();
        goto start;
        break;

    case '2':
        system ("cls");
        search();
        goto start;
        break;

```

```

case '3':
    system ("cls");
    del();
    goto start;
    break;
case '4':
    system ("cls");
    Exit();
    break;
default:
    printf("Invalid choice.");
    system ("cls");
    goto start;
}
getch();
return 0;
}

void test()
{
    FILE *fp;
    int a[20];
    printf("Enter your first name:");
    scanf("%s",b.namef);
    printf("Enter your last name:");
    scanf("%s",b.namel);
    printf("Enter your email:");

```

```

scanf("%s",b.email);

printf("Enter your Contact no.:");

scanf("%s",b.contact);

system("pause");

system ("cls");

printf("Instructions\n");

printf("The CES-D should take no more than 10 minutes to complete.\n");

printf("The provided statements about the depression are to be answered on how
often you experience them(over a week) by entering a number in the blank after each
statement, according to the following scale:\na. 0 = Rarely or none of the time (less
than 1 day)\nb. 1 = Some or a little of the time (1-2 days)\nc. 2 = Occasionally or a
moderate amount of time (3-4 days)\nd. 3 = Most or all of the time (5-7 days)\n");

system("pause");

printf("\nPlease read the statements carefully, some of the statements(d, h, l and p)
indicate positive traits(p) for which the following scale applies: \na. 3 = less than 1
day\nb. 2 = 1-2 days \nc. 1 = 3-4 days \nd. 0 = 5-7 days\n");

system("pause");

printf("\nDo not take too long over individual questions; there are no \"right\" or
\"wrong\" answers (and no trick questions). The first answer that comes into your
head is probably the right one for you. If you find some of the questions difficult,
please give the answer that is true for you in general or for most of the time. The
obtained score will be discarded, when more than four statements remained
unanswered or numbers other than the above mentioned are entered.\n");

system("pause");

printf("\nThe range of possible scores falls between 0-60.");

printf(" A cutoff score of 16 or greater reflects individuals at risk for clinical
depression. More specifically:\n0-16: No to mild depressive symptomatology\n16-23:
Moderate depressive symptomatology\n24-60: Severe depressive
symptomatology\n");

system("pause");

system("cls");

printf("Statements:\n");

```

```

printf("a.I was bothered by things that usually don't bother me.\n");
scanf("%d",&a[0]);
printf("b.I did not feel like eating; my appetite was poor.\n");
scanf("%d",&a[1]);
printf("c.I felt that I could not shake off the blues even with help from my family
and friends.\n");
scanf("%d",&a[2]);
printf("d.I felt that I was just as good as other people(p).\n");
scanf("%d",&a[3]);
printf("e.I had trouble keeping my mind on what I was doing.\n");
scanf("%d",&a[4]);
printf("f.I felt depressed.\n");
scanf("%d",&a[5]);
printf("g.I felt that everything I did was an effort.\n");
scanf("%d",&a[6]);
printf("h.I felt hopeful about the future(p).\n");
scanf("%d",&a[7]);
printf("i.I thought my life had been a failure.\n");
scanf("%d",&a[8]);
printf("j.I felt fearful.\n");
scanf("%d",&a[9]);
printf("k.My sleep was restless.\n");
scanf("%d",&a[10]);
printf("l.I was happy(p).\n");
scanf("%d",&a[11]);
printf("m.I talked less than usual.\n");
scanf("%d",&a[12]);
printf("n.I felt lonely.\n");

```

```

scanf("%d",&a[13]);

printf("o.People were unfriendly.\n");

scanf("%d",&a[14]);

printf("p.I enjoyed life(p).\n");

scanf("%d",&a[15]);

printf("q.I had crying spells.\n");

scanf("%d",&a[16]);

printf("r.I felt sad.\n");

scanf("%d",&a[17]);

printf("s.I felt that people disliked me.\n");

scanf("%d",&a[18]);

printf("t.I could not get \"going\".\n");

scanf("%d",&a[19]);

b.score=0;

for(int i=0; i<=19; i++)
{
    b.score = b.score + a[i];
}

system ("cls");

if (b.score<=16)
{
    printf("\nYour Depression index is %d which reflects No to Mildly depressive
symptomatology.\n",b.score);
}

else if(b.score>16 && b.score<=23)
{
    printf("\nYour Depression index is %d which reflects Moderately depressive
symptomatology.\n",b.score);
}

```



```

else if(b.score>23 && b.score<=60)
{
    printf("\nYour Depression index is %d which reflects Severely depressive
symptomatology.\n",b.score);
}
else{
    printf("Invalid score.");
}

system("pause");

printf("\n\nNote: This test is scored so that higher scores indicate greater symptoms
of depression. Before interpreting your score, you should know that a high score is not
the same thing as a diagnosis of depression. Some people who get high scores are not
in fact depressed, and people with low scores can still have a \"depressive disorder\".
A full-blown diagnosis of depression depends on other things, such as how long your
symptoms have lasted and whether they have some primary source other than
depression. A diagnosis can be made only after a thorough interview with a qualified
psychologist or psychiatrist.\n");

system("pause");

fp = fopen("test.txt","a");
if(fp==NULL)
{
    printf("\nFile cannot be created.");
    exit(1);
}

fprintf(fp, "\n%s %s %s %s %d", b.namef,b.namel,b.email,b.contact,b.score);
fclose(fp);
}

void search()
{
char san[20];

```

```

start:
system("cls");
printf("Enter Name: ");
scanf("%s",san);
FILE *fp1;
fp1= fopen("test.txt", "r");
if(fp1==NULL)
{
printf("\nFile cannot be created.");
exit(1);
}
while((fgetc(fp1)) !=EOF)
{
fscanf(fp1,"%s",b.namef);
fscanf(fp1,"%s",b.namel);
fscanf(fp1,"%s",b.email);
fscanf(fp1,"%s",b.contact);
fscanf(fp1,"%d",&b.score);
if(strcmp(san,b.namef) == 0)
{
printf("\nDetails:");
printf("\nName: %s %s \nEmail id: %s\nContact no.: %s\nDepression index:
%d\n",b.namef,b.namel,b.email,b.contact,b.score);
system("pause");
goto begin;
}
}
if(strcmp(san,b.namef) == 1)
{

```

```

printf("\nThis name does not exist.");

begin:
printf("\nDo you want to search again[Yes(y)/No(n)]?");
scanf("\n%c",&ch);
if(ch == 'y')
{
goto start;
}
else if(ch == 'n')
{
goto end;
}
else{
printf("\nInvalid choice.");
goto begin;
}
}
end:
fclose(fp1);
}
void del()
{
char sun[20];
start:
system("cls");
printf("Enter the Testee's name whose record is to be deleted: ");
scanf("%s",sun);
FILE *fp2, *fp3;

```

```

fp3 = fopen("temp.txt","w");
if(fp3==NULL)
{
    printf("\nFile cannot be created.");
    exit(1);
}
fp2 = fopen("test.txt","r");
if(fp2==NULL)
{
    printf("\nFile cannot be created.");
    exit(1);
}
int d=0;
while((fgetc(fp2)) !=EOF)
{
    fscanf(fp2,"%s",b.namef);
    fscanf(fp2,"%s",b.namel);
    fscanf(fp2,"%s",b.email);
    fscanf(fp2,"%s",b.contact);
    fscanf(fp2,"%d",&b.score);
    if(strcmp(sun,b.namef) == 1)
    {
        fprintf(fp3, "\n%s %s %s %s %d",
b.namef,b.namel,b.email,b.contact,b.score);
    }
    if(strcmp(sun,b.namef) == 0)
    {
        d=1;
    }
}

```

```

}
if(d == 0)
{
    printf("\nThis name does not exist.");
    begin:
    printf("\nDo you want to try it again[Yes(y)/No(n)]?");
    scanf("\n%c",&ch);
    if(ch == 'y')
    {
        goto start;
    }
    else if(ch == 'n')
    {
        goto end;
    }
    else{
        printf("\nInvalid choice.");
        goto begin;
    }
}
end:
fclose(fp2);
fclose(fp3);
remove("test.txt");
rename("temp.txt","test.txt");
if (d==1)
{
    printf("\n Deleted successfully.\n");

```

```

system("pause");
    }
}
void Exit()
{
    gotoxy(20,8);
    printf("*****THANK YOU FOR PARTICIPATING*****\n");
}
void gotoxy(int x, int y)
{
    COORD c;
    c.X=x;
    c.Y=y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),c);
}

```