EXPERIMENT-7 DATE: 16/10/2021

⇒ **AIM:-** A farmer has a field in the shape of a rhombus. The perimeter of the field is 400m and one of its diagonal is 120m. He wants to divide the field into two equal parts to grow two different types of vetegables. Draw the flow chart to find the area of the field.

⇒ **ALGORITHM:-**

Step 1: Start

Step 2: Side = 100

Step 3: Set diagonal 1 = 120

Step 4: Calculate half of diagonal 1, hd1 = 120/2

Step 5: Calculate half of diagonal 2, ~~hd2~~

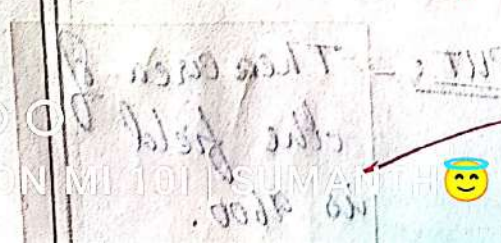
$$hd_2 = \sqrt{side^2 - hd_1^2}$$

Step 6: Set $d_2 = 2 \times hd_2$

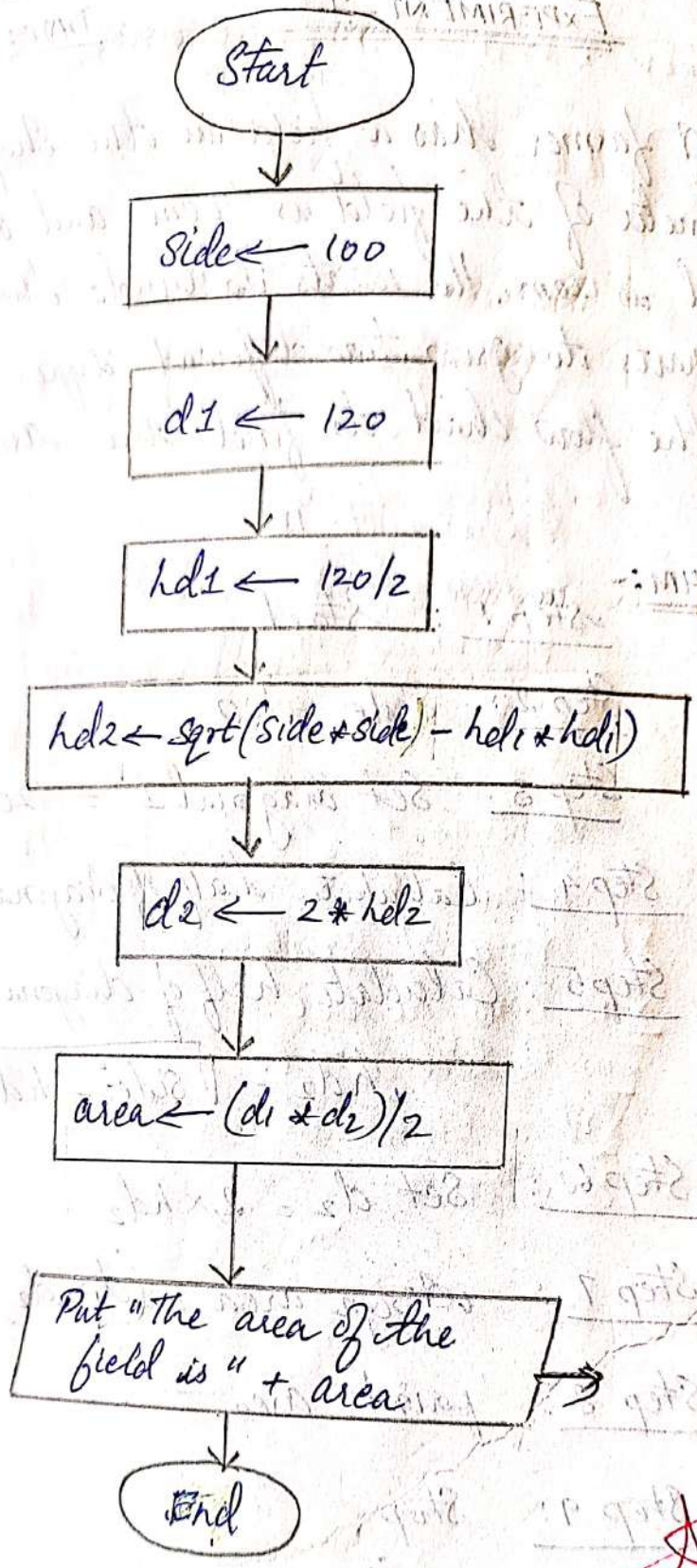
Step 7: Assign area = $(d_1 * d_2)/2$

Step 8: print area.

Step 9: Stop.



⇒ FLOWCHART :-



⇒ INPUT :- side = 100
hd1 = 60
hd2 = 80
d1 = 120
d2 = 160

⇒ OUTPUT :- The area of the field is 9600.

⇒ RESULT :- Hence, the program is executed successfully.

EXPERIMENT – 11 DATE: 16/10/2021

⇒ AIM :- Draw the flowchart to check whether it is raining ~~out~~ or not. If it is, then open the umbrella or wear the cap.

⇒ ALGORITHM :- Step 1 : Start

Step 2 : Ask if it is raining.

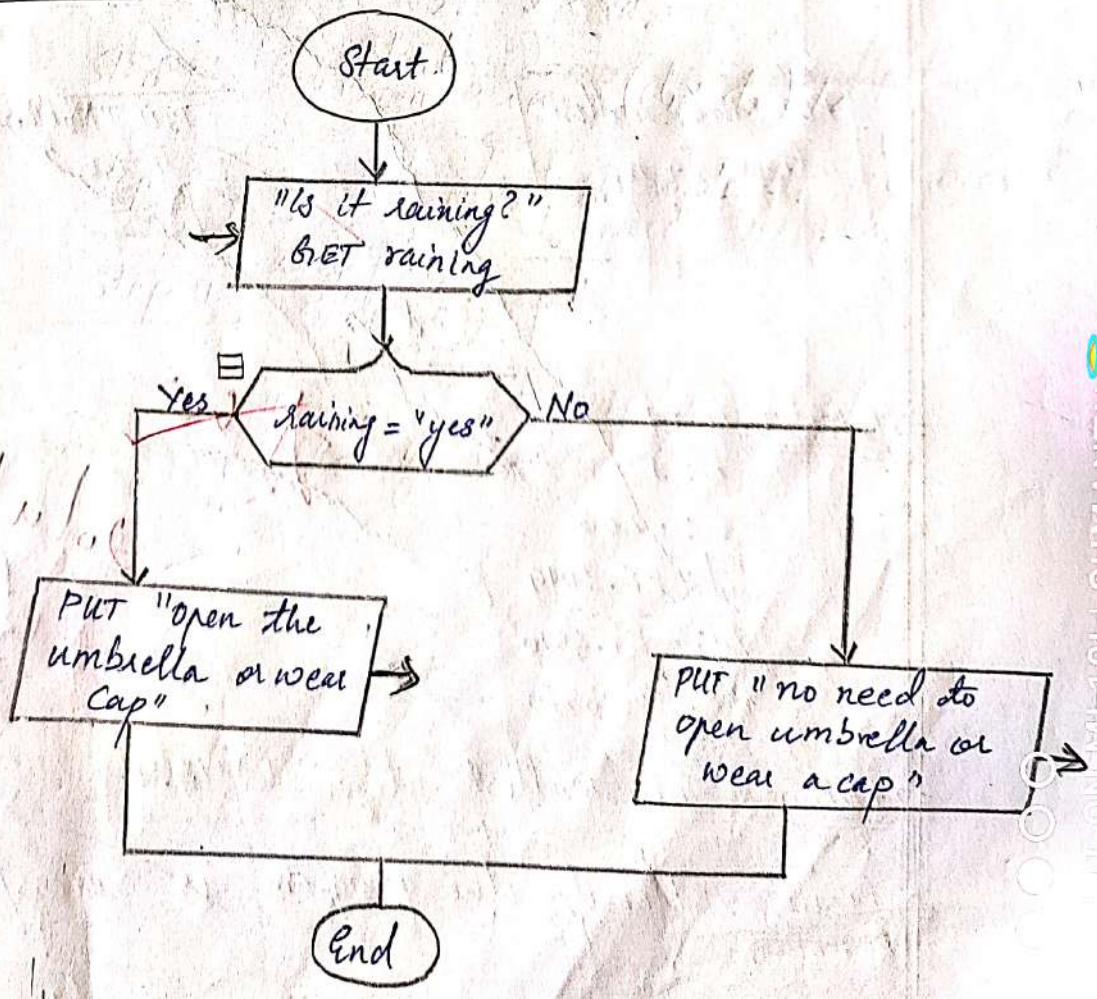
Step 3 : If raining = yes then go to step 4 and if raining = no then go to step 5.

Step 4 : Print "open the umbrella or wear a cap."

Step 5 : Print "no need to wear cap or use an umbrella."

Step 6 : Stop.

⇒ FLOWCHART :-



⇒ Input :- raining = "yes"

⇒ Output :- Open the umbrella or wear a cap.

⇒ RESULT :- Hence, the program is executed successfully.

20/4/21

EXPERIMENT - III

DATE = 23/10/2021

⇒ **AIM :-** Draw the flowchart to find the smallest integer n such that $1+2+3+\ldots+n$ is equal to or just greater than 100.

⇒ **ALGORITHM :-**

Step 1: Start

Step 2: assign a = 1, b = 1 c = -200

($n^2+n-200=0$)

Step 3: Find the square roots using quadratic formula.

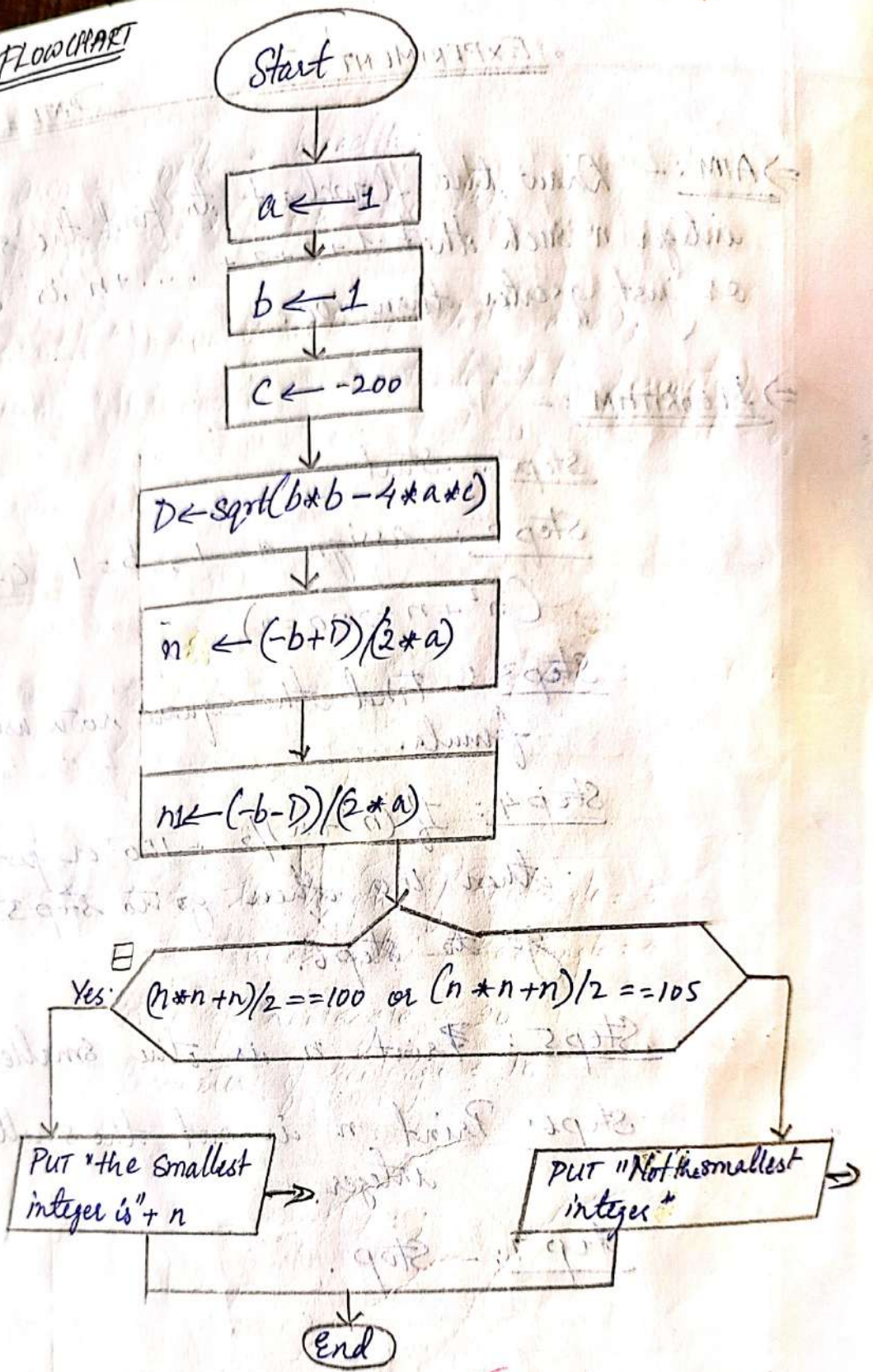
Step 4: If $(n^2+n)/2$ = 100 or just greater than 100 then go to step 5 otherwise go to step 6.

Step 5: Print n is the smallest integer.

Step 6: Print n is not the smallest integer.

Step 7: - Stop.

FLOWCHART



⇒ Input: a : 1
b : 1
c : -200
d : 28.3019
n1 : -14.6510
n : 13.6510

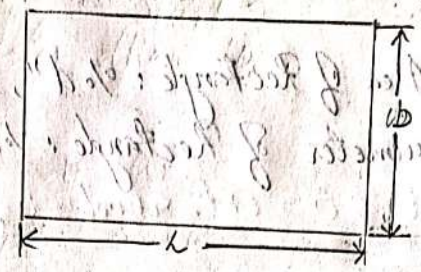
⇒ Output: The smallest integer is 14.

⇒ RESULT : Hence, the program is successfully executed.

EXPERIMENT - IV

DATE:- 30/10/2021

⇒ **AIM :-** Design and develop a C program to find area and perimeter of rectangle. P = 2(l+b).



⇒ **ALGORITHM :-** **Step 1** :- Declare the variables length, breadth, area and perimeter.

Step 2 :- Take length and width as input from user using scanf function and it gets stored in length and breadth variables.

Step3 : Multiply length by breadth to find area and the value is stored in "area".

Step4 : Multiply 2 * (l+b) to find the perimeter and the value is stored in the variable "perimeter".

Step5 : Print the ~~value~~ area of the rectangle and perimeter of the rectangle using printf.

⇒ **PROGRAM CODE:-**

```
#Include <Stdio.h>
int main() {
int length, breadth, area, perimeter;
Printf("\nEnter the length of rectangle: ");
scanf("%od", &length);
Printf("\nEnter the Breadth of Rectangle: ");
Scanf("%d, &breadth);
```

```
area = length * breadth;
perimeter = 2*(length + breadth);

printf("\nArea of Rectangle : %d", area);
printf("\nperimeter of Rectangle : %d", perimeter);

return (0);
}
```

⇒ Input :- 5
4

⇒ Output :-

```
Enter the length of Rectangle = 5
Enter the breadth of Rectangle = 4
Area of Rectangle = 20
Perimeter of Rectangle = 18
```

20/11/21

⇒ Result :- The program was executed successfully.

EXPERIMENT - V

DATE:- 24/10/2021

=> AIM: Write a C program to read an age of 15 person and find out how many of them fall under :-

(a) still a baby - age 0 to 5
(b) Attending school - age 6 to 17
(c) Adult life - age 18 and over

(Using while loop)

=> ALGORITHM:

Step 1 :- Start

Step 2: Declare integers age, cnt_baby, cnt_school, cnt_adult, and count.

Step 3: add value zero for cnt_baby, cnt_school and cnt_adult and ~~ent~~ count.

Step 4: Use while loop.

Step 5: Print "Enter the age of person". and 1 to integer count.

Step 6: Use scanf to take the input from the user.

Step 7: Using if else ... if statement check the age of the person and which category they belong to.

Step 8: Display their age and category using printf.

=> PROGRAM CODE :-

```
#include <stdio.h>
int main()
{
    int age;
  int cnt_baby = 0 , cnt_school = 0, cnt_adult = 0;
  uint count=0;
 while (count<15)
  {
    printf ("Enter age of person [%d]: ", count+1);
    Scanf ("%", &age);

    if (age>=0 && age<=5)
          cnt_baby++;
    else if (age>=6 && age<=17)
          cnt_school++;
     else
          cnt_adult++;
    //increase counter
    count++;
  }

  printf ("Baby age : %d \n", cnt_baby);
  printf ("School it age : %d \n ", cnt_school);
  printf ("Adult age : %d \n", cnt_adult);
  return 0;
  }
```

⇒ INPUT :-

Enter age of person [1] = 20
Enter age of person [2] = 15
Enter age of person [3] = 6
Enter age of person [4] = 36
Enter age of person [6] = 3
Enter age of person [7] = 8
Enter age of person [8] = 13
Enter age of person [9] = 65
Enter age of person [10] = 25
Enter age of person [11] = 10
Enter age of person [12] = 17
Enter age of person [13] = 5
Enter age of person [14] = 19
Enter age of person [15] = 44
Enter age of person [15] = 1

⇒ Output :-

```
Baby age = 3
School age = 6
Adult age = 6
```

20/4/21

⇒ RESULT :- Hence, the C program to check the age and category of the 15 people was executed successfully.

EXPERIMENT-6

19-11-2021

⇒ **AIM:-** Design and develop a 'C' program to find whether a given number is armstrong number or not.

⇒ **ALGORITHS:-**

Step 1: Start

Step 2: Declare integers n, r, sum=0 and temp.

Step 3: Take input from the user using scanf.

Step 4: Put temp = n

Step 5: Using while loop check if the number is an armstrong number.

Step 6: Use if else loop

Step 7: print the number as armstrong number if it is calculated to be otherwise print it as not an armstrong number.

⇒ **PROGRAM CODE:-**

```
#include<stdio.h>
int main()
{
  int n, r, sum=0, temp;
  printf("Enter the number = ");
  scanf("%d", &n);
  temp = n;
  while(n>0)
  {
  r = n%10;
  sum = sum + (r*r*r);
  n = n/10;
  }
```

```
if (temp == sum)
printf("armstrong number");
else
printf("not armstrong number");
return 0;
}
```

=> INPUT: 15 ↵
371 ↵

=> OUTPUT:

Not armstrong number
Armstrong number

20/11/21

=> RESULT: Hence, Designed and developed a C program to calculate and check if the given number is an armstrong number. Executed successfully.

EXPERIMENT - 7 17-12-2021

⇒ Aim :- Write a Program in C to find the second largest bill amount in number of customers in the super...

⇒ ALGORITHM :- Step 1 : Start

Step 2 : Declare 1 one dimensional array a[] of size 50.

Step 3 : Declare int variable i to iterate the array elements.

Step 4 : Take input from the user for array a[] size and assign the user entered values to size variable.

Step 5 : Take input for entering the bill amount.

Step 6 : Use for loop to iterate each cell present in a[50] array.

Step 7 :- Use scanf statement inside the for loop to store the user entered values into individuals array element such as a[0], a[1] ....

Step 8 :- Use another for loop to iterate each element in an array.

Step 9 :- Use if statement inside the for loop to check whether a[i] is equal to search item or not.

Step 10 :- Print the second largest bill amount using printf.

Step 11 :- Stop.

⇒ PROGRAM CODE:-

```
#Include <stdio.h>
  Void main()  (Change to int main when doing it
                on online gdb.
  {
  int a[50]
  Int n, i, max, s_large;
  printf("\n Enter the number of customers in supermarket.");
  scanf("%d", &n);
  printf("\n Enter the bill amount");
  for(i=0; i<n; i++)
    {
    scanf("%d", &a[i]);
    }
  max = s_large = a[0];
  for(i=1; i<n; i++)
    {
    if(max<a[i])
    {
    s_large = max;
      max = a[i];
    }
    else if(s_large<a[i] && a[i] != max)
      {
      s_large = a[i];
      }
  }
  Print("\n The second largest bill amount: %", s_large);
  }
```

⇒ Input :- 6

22 23 100 43 65 11

65

⇒ OUTPUT :-

```
Enter the number of customers in supermarket = 6
Enter the bill amount : 22 23 100 43 65 11
The second lagest bill amount : 65
```

Ahi

⇒ RESULT :- Hence, designed and developed a C program to find the second largest bill amount & number of customers in supermarket.
Executed successfully.

Experiment- 8

⇒ AIM:- Write a program to read a matrix of size r x c and print its transpose.

⇒ ALGORITHM :- Step 1 :- Start

Step 2 :- Declare and initialize a 2D array a[r][c] of order r x c.

Step 3: Read the matrix a[r][c] from the user, store the transpose of the matrix. This array will have the reversed dimensions of the original matrix.

Step 5: The next step is to loop through the original array and then convert its rows to the columns of matrix transpose.

Step 4 :- Declare another 2-dimensional array transpose[] to store the transpose of the matrix. This array will have the ~~reversed~~ reversed dimensions of the original matrix.

5.1 :- Declare 2 variables i and j

5.2 : Set both i, j = 0

5.3 : Repeat until i < a

5.4 : and repeat until j < c

5.5 : transpose[j][i] = a[i][j]

5.6 :- j = j+1

5.7 : i = i+1

Step 6 :- Display the elements of the transposed matrix transpose.

Step 7 :- Stop (END)

=) PROGRAM CODE:-

```
#include <stdio.h>
int main()
{
int a[10][10], transpose[10][10], r, c;
printf("Enter rows and columns: ");
scanf("%d %d", &r, &c);
printf("\nEnter matrix elements: \n");
for(int i = 0; i<r; ++i)
for(int j = 0; j<c; ++j)
{
printf("Enter element a%d%d: ", i+1, j+1);
scanf("%d", &a[i][j]);
}
printf("\nEntered matrix: \n");
for(int i = 0; i<r; ++i)
for(int j = 0; j<c; ++j)
{
printf("%d", a[i][j]);
if(j == c-1)
printf("\n");
}
for(int i = 0; i<r; ++i)
for(int j = 0; j<c; ++i)
{
transpose[j][i] = a[i][j];
}
printf("\n Transpose of the matrix: \n");
for(int i = 0; i<c; ++i)
```

```
for(int j = 0; j<r; ++j)
{
printf("%d", transpose[i][j]);
if(j==r-1)
printf("\n");
}
return 0;
}
```

⇒ INPUT: - 3 3

4

5

6

7

8

9

1

3

5

⇒ OUTPUT: - Enter rows and columns: 3 3

Enter matrix elements:

Enter element a_{11}: 4

Enter element a_{12} = 5

Enter element a_{13} = 6

Enter element a_{21} = 7

Enter element a_{22} = 8

Enter element a_{23} = 9

Enter element a_{31} = 1

Enter element a_{32} = 3

Enter element a_{33} = 5

Entered Matrix:

```
4   5   6
7   8   9
1   3   5
```

Transpose of the Matrix:-

```
4   7   1
5   8   3
6   9   5
```

fn

⇒ **RESULT:-** Hence, the C program to read 2×2 matrix and find its transpose was executed successfully.

Experiment -9

⇒ **AIM:-** Write a C program to find maximum and minimum of an array of n elements using functions.

⇒ **ALGORITHM:-** **Step 1:-** Start

Step 2: We take an integer array having number as Arr[].

Step 3:- The function getResult(int arr[], int n) is to find the maximum and minimum element present in the array in minimum no. of comparisons.

Step 4:- If there is only one element then we will initiate the variable max and min with arr[0].

Step 5:- For more than one element, then we will ~~initiate~~ initialize max with arr[1] and min with arr[0].

Step 6:- Inside for loop start traversing from third element (i=2) till last.

Step 7:- Now, we will compare each value (arr[i]) with min and max. If its less than min, update min with arr[i]. If it is more than max then update max with arr[i]

Step 8:- In the end, print the results stored in max and min variable.

Step 9: Stop (END).

⇒

⇒ Program Code :-

```
#include <stdio.h>
#include <math.h>
int getresult(int arr[], int n)
{
  int min = 0, max = 0;
/* If there is only one element then return it as min and
max both */
   if(n==1)
{ min = max = arr[0];
 }
/* If there are more than one elements, then initialize min and
max */
 if(arr[0] > arr[1])
   { max = arr[0];
   min = arr[1];
   }
else
{max = arr[1];
   min = arr[0];
 }
for(int i = 2 ; i<n; i++)
   {
    if(arr[i] > max)
    max = arr[i];
    else if(arr[i] < min)
      min = arr[i];
    }
 printf("Minimum element: %d", min);
 printf("Maximum element: %d", max);
}
```

/* Driver program to test above function */

```
int main()
{
    int arr[] = {200, 191, 112, -11, 330, 60};
    int n = 6;
    getResult(arr, n);
}
```

⇒ INPUT :- Enter size of the array : 5
Enter 5 elements in array : 2 3 6 9 1

⇒ OUTPUT :-

Enter size of the array : 5
Enter 5 elements in array : 2 3 6 9 1
Minimum element in array = 1
Maximum element in array = 9

⇒ RESULT :- Hence, the C program was executed successfully.

Experiment - 10

⇒ AIM :- Develop a C program to read the name, Roll no and Stream of the student. Display all details of the student.

⇒ ALGORITHM :-

Step 1 : Start

Step 2 :- Create a structure with fields Roll, name and Stream.

Step 3 : Read the students records in the structure.

Step 4 :- Define a comparator by setting up rules for comparison. Here stream can be sorted with the help of "difference" of the stream of 2 students (Students 1 → Student 2 → stream

Step 5 :- Now sort the structure based on the defined comparator with the help of "qsort()" method.

Step 6 :- Print the sorted students "records".

Step 7 :- Stop.

⇒ PROGRAM CODE:-

```
#include <stdio.h>
typedef struct {
 char name[30];
 int roll;
 Char stream[30];
} Student;
int main ()
{
Char buffer;
 int n=3;
 Student students[n];
 printf("Enter %d student Details \n\n",n);
 for (int i=0; i<n; ++) {
  Printf("Student %d :- \n", i+1);
  printf("Name");
  Scanf("%[^\n]", students[i].name);
  Scanf("%c", &buffer);
  Printf("Roll:");
  Scenf("%d", &students[i].roll);
  Scanf("%c", &buffer);

 Printf("Stream:");
 Scanf("%[^n]", students[i].stream);
  Scanf("%c", &buffer);

 printf("\n");
 }
printf("------All students Details ------\n");
for(int i=0; i<n; i++)
 {
  printf("Name-😇");
```

```
printf("%s\n", students[i].name);
printf("Roll\t: ");
printf("%d\n", students[i].roll);
printf("Stream\t: ");
printf("%s\n", students[i].stream);
printf("\n");
}
return 0;
}
```

⇒ INPUT :-

```
Afrina
2
Science

Yash
39
CS

Devika
5
Aeronautics
```

⇒ OUTPUT :-

```
Enter 3 student Details

Student 1:-
Name: Afrina
Roll: 02
Stream:- Science

Student 2:-
Name:- Yash
Roll:- 39
Stream:- CS

Student 3:-
Name:- Devika
Roll:- 5
Stream:- Aeronautics
```

-------- All Students Details ----------

Name: Afrina
Roll: 02
Stream: Science

Name: Yash
Roll: 39
Stream: CS

Name: Devika
Roll: 05
Stream: Aeronautics

⇒ RESULT:- Hence, the C program was executed successfully.

EXPERIMENT - 11

⇒ **AIM:-** Write a code in C to compute internal marks of student for five different subjects using array of structures.

⇒ **ALGORITHM:-**

Step 1 :- start the program

Step 2 :- Declare the structure student with variables name, roll no., mark, total.

Step 3 :- Declare the necessary variables.

Step 4 :- Create a structure variable using arrays.

Step 5: Get the student roll nos, name, mark using for loop.

Step 6: Calculate the total marks using arithmetic operator.

Step 7:- Using for loop display name, roll no, and total for each student.

Step 8 :- Stop the program.

⇒ **PROGRAM CODE:-**

```
#include <stdio.h>
int main()
{
float subject_1, subject_2, subject_3, subject_4, subject_5;
float total, average;
printf("Enter the marks of five subjects :: \n");
scanf("%f %f %f %f %f", &subject_1, &subject_2, &subject_3,
&subject_4, &subject_5);
//It will calculate the total, average
total = subject_1 + subject_2 + subject_3 + subject_4 +
subject_5;
```

```
average = total/5.0;
// It will produce the final output
printf("\n The total marks is: \t%.2f/500.00\n", total);
printf("\n The average marks is: \t%.2f\n", average);
return 0;
}
```

⇒ Input:- 98 97 97 96 95

⇒ Output:- Enter the marks of five subjects:-

98
97
97
96
95

The total marks is: 483.00/500.00

The average marks is: 96.00

⇒ RESULT:-

Hence, the c program was executed successfully.

Experiment - 12.

⇒ **AIM:** Write a program in C to create a text file named "line.txt" and display the number of words, number of characters and number of lines in a text file.

⇒ **ALGORITHM :-**

Step 1 : The main() calls the stringcount() function by passing the character array(string) as an argument.

Step 2 :- The function stringcount() counts the no. of alphabets, digits and special characters present in the given string as follows for loop iterates through the string until s[i] becomes null.

(a) Increase the alphabet count if the ASCII value of s[i] is the range between 65 to 90 or 97 to 122.

(b) Increase the digits count if the ASCII value of s[i] is in the range between 48 ~~and~~ to 57.

(c) If s[i] is not in range of the above two conditions then increase the special characters count.

Step 3 :- After all iterations of for loop the function prints the values of alphabets count, digits count and Special characters count.

⇒ **PROGRAM CODE :-**

```
#include<String.h>
void stringcount(char *s)
  {
  int
  i, alphabets = 0, digits=0, Specialcharacters = 0,

  for(i=0; s[i]; i++)
  {
  if((s[i]>=65 && s[i]<=90) || (s[i]>=97 && s[i]<=122))
      alphabets++;
  else if(s[i]>=48 && s[i]<=57)
      digits++;
```

```
else
Specialcharacters++;
}
printf("Alphabets = %d\n", alphabets);
printf("Digits = %d\n", digits);
printf("Special Characters = %d", specialcharacters);
}
int main()
{
Char s[1000];
printf("Enter the string");
gets(s);
Stringcount(s);
}
```

⇒ Input :- 12Harry_potter#

⇒ Output :- Enter the string: 12Harry_potter#
Alphabets: 11
Digits: 2
Special Characters: 2

⇒ RESULT :- Hence, C program was executed successfully.

EXPERIMENT - 13

* **AIM :-** Design and develop a C program to search a particular product detail from given set of product record which is to be stored in a binary file with product Id in sorted order.

* **ALGORITHM :-**

Step 1 :- Start

Step 2 :- T(N) is the time complexity of the binary search for a set of N elements. Then

$T(N) = T(N/2) + O(1)$ (By means of the recurrence relation) -(i)

Step 3 :- Now, applying Masters Theorem for computing run time complexity of recurrence relation ie,

$T(N) = aT(N/b) + f(N)$ — (ii)

Comparing equation (ii) with (i), we get,

$a = 1, b = 2$

Hence, $\log(a \text{ base } b) = 1 = c$ — (iii)

Step 4: Now, $f(N) = n^c \log^k(n)$ // $k = 0$ — iv

Using (iii) and (iv) in equation (ii), we get,

$T(N) = O(N^c \log^{(k+1)}(N)) = O(\log(N))$ — (v)

Step 5 :- Stop

* **PROGRAM CODE :-**

```
#include <stdio.h>
int main()
{ int c, first, last, middle, n, search, array[100];
printf("Enter number of elements:\n");
scanf("%d", &n);
printf("Enter %d integers:\n", n);
```

```
for (c= 0; c<n; c++)
  scanf("%d", &array[c]);
printf("Enter the value to find: \n");
  scanf("%d", &search);
first = 0;
last = n - 1;
middle = (first + last)/2;
while(first <= last)
{
  if(array[middle] < search)
        first = middle + 1;
  else if(array[middle] == search)
   {
   printf("%d is present at index %d.\n", search, middle+1
   break;
  }
else
   last = middle - 1;
   middle = (first + last)/2;
  }
if (first>last)
 printf(" Not found! %d is not present in the list .\n", search);
 return 0 ;
}
```

* INPUT:- 3

235 55 12

55

⇒ **OUTPUT:-**

```
Enter number of elements:
3
Enter 3 integers:-
235 55 12
Enter the value to find:
55
55 is present at index 2.
```

⇒ **RESULT :-**

Hence, the C program was executed successfully.

EXPERIMENT - 13

* AIM :- Write a graphics program to create a smiley face or any image of your choice.

* ALGORITHM :-

Step 1 : Start

Step 2 : graphics.h is used to make graphics in C language.

Step 3 : first thing is to initiate a graph. This will allow us to make graphics.

Step 4 : Second thing is to set the color to yellow.

Step 5 : With the help of circle() function create a circle by passing the parameters for a center with radius.

Step 6 : Fill the circle with yellow colours with the help of setfillstyle() and floodfill().

Step 7 : By calling the function fillellipse(), make 4 ellipses with the colour black.

Step 8 : Stop the program.

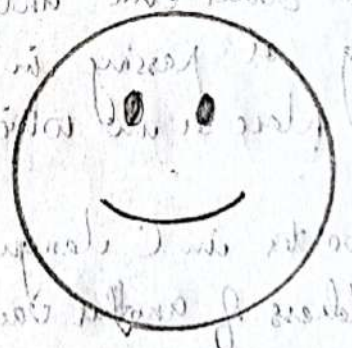
* PROGRAM CODE :-

```
#include <graphics.h>
#include <conio.h>
int main()
{ int graphicdriver = DETECT, graphicmode;
  initgraph(&graphicdriver, &graphicmode, "C:\\turboc3\\bgi");
  outtextxy(10,10, "Program to draw a smiley face in
  C graphics");
```

```
setcolor(YELLOW);
circle(300,100,40);
setfillstyle(SOLID_FILL, YELLOW);
floodfill(300,100, YELLOW);
setcolor(BLACK);
setfillstyle(SOLID_FILL, BLACK);
fillellipse(310, 85, 2, 6);
fillellipse(290, 85, 2, 6);
ellipse(300, 100, 205, 335, 20, 9);
ellipse(300, 100, 205, 335, 20, 10);
ellipse(300, 100, 205, 335, 20, 11);
getch();
return 0;
}
```

⇒ **OUTPUT:**

Program to draw a smiley face in C graphics.



⇒ **RESULT:**

Hence, the graphics program in C was executed successfully.