Ex.No.1 13-09-23	TECHNICAL PROBLEMS AND FLOWCHART	Reg.No:- URK23CS1261

1a)

Aim: The objective of this activity is to develop a Flowchart for Electricity Billing.

Algorithm

Step 1 : Start

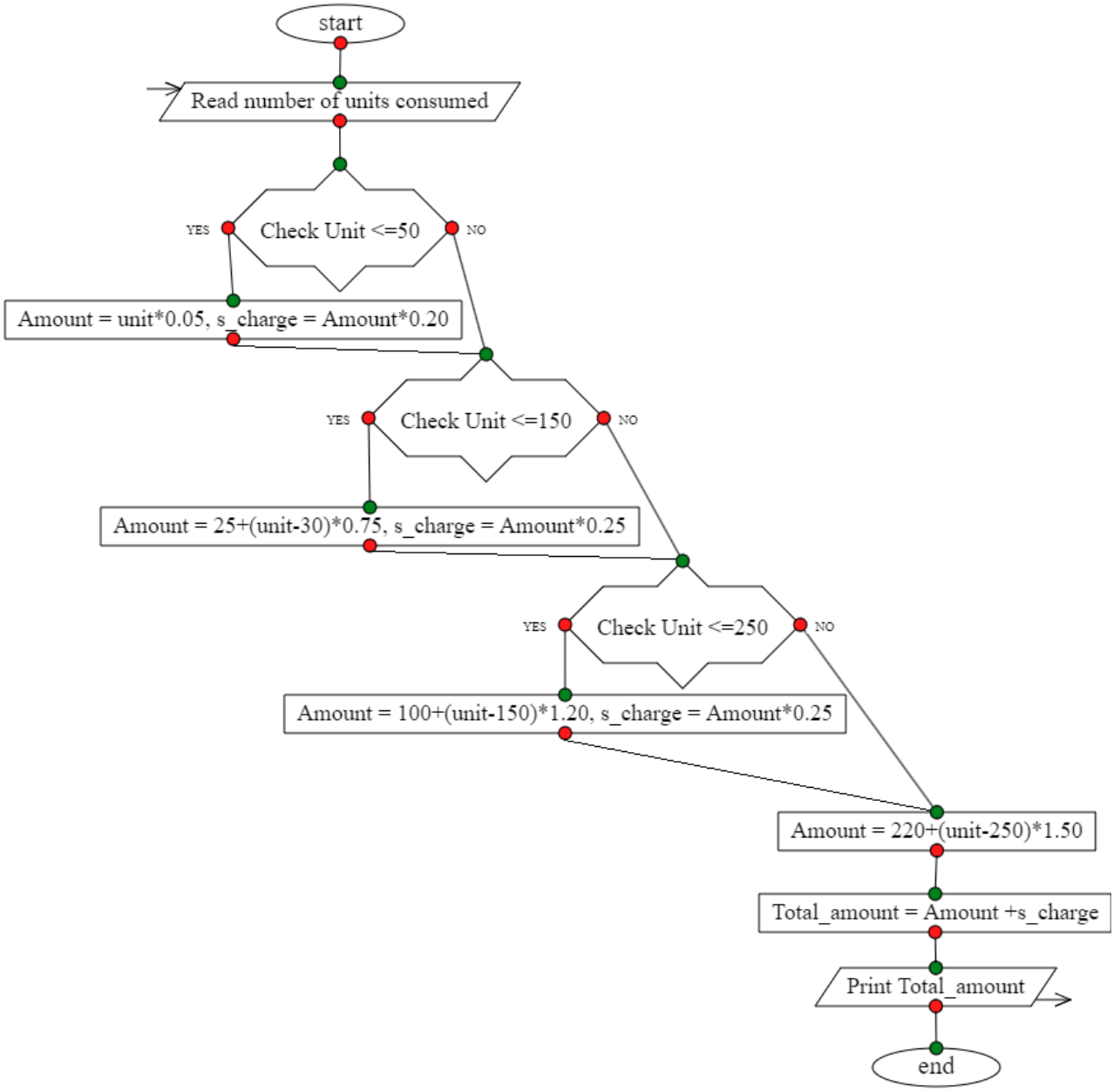
Step 2 : Read the previous unit and current unit

Step 3 : Calculate the used units by subtracting the current unit and previous unit

Step 4 : Calculate the Electricity bill from the used units

Step 5 : Print the amount of Electricity Bill

Step 6 : Stop



Result:

The algorithm has successfully depicted the process behind the electric billing system for the suitable input and desired output.

1b)

Aim: The objective of this activity is to develop a Flowchart Retail Shop Billing System

Algorithm

Step 1 : Start

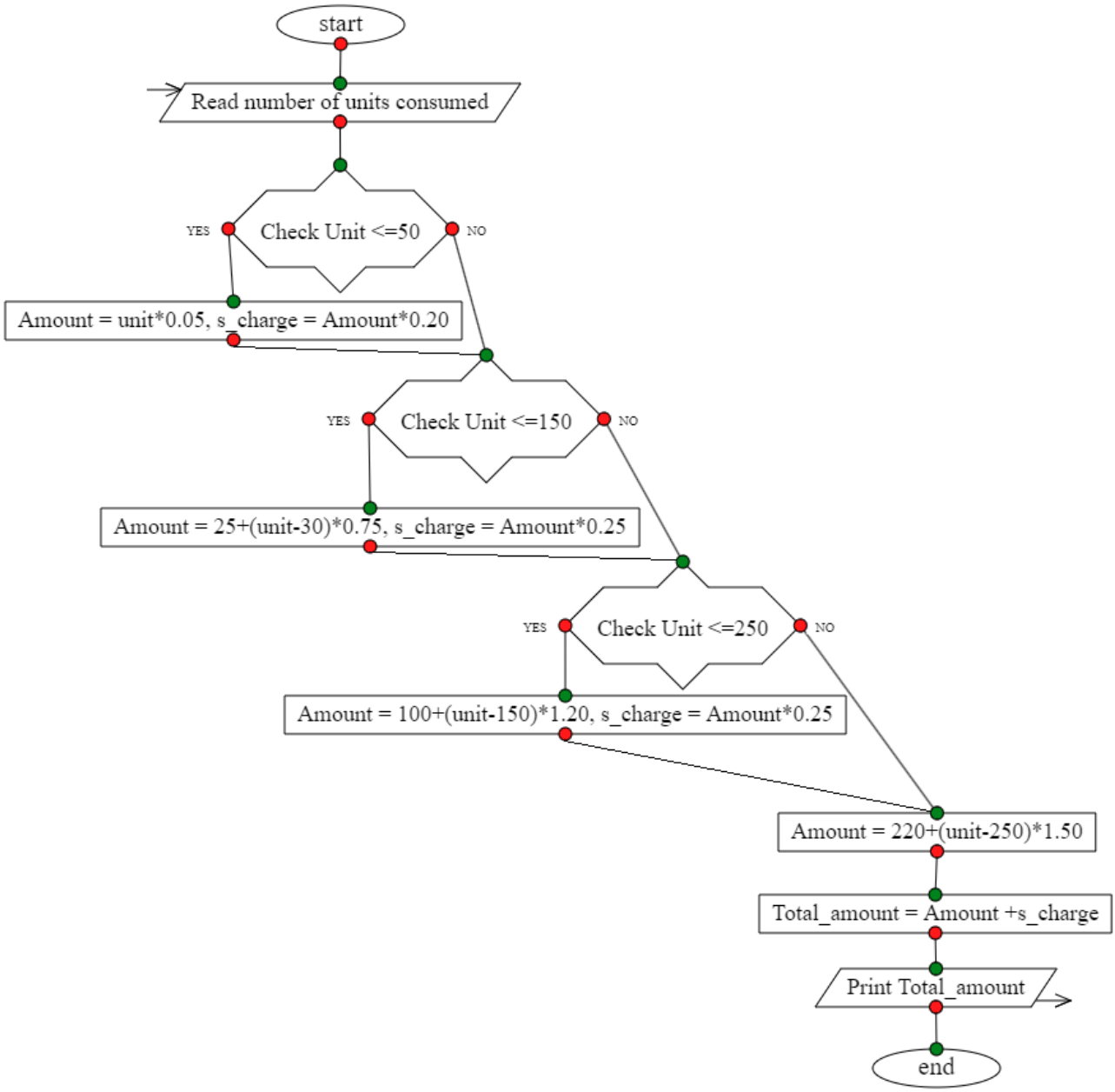
Step 2 : Read the barcode of the product

Step 3 : Display the product name and amount

Step 4 : Check if more products available, if available go to step 2, otherwise go to step 5 Step Step 5 : Calculate the total cost of the products

Step 6 : Print the total cost

Step 7 : Stop



Result:

The algorithm has successfully depicted the process behind the retail shop billing system for the suitable input and desired output.

1c)

Aim : The objective of this activity is to develop a Flowchart, an electric current calculation in 3 phase AC circuit.

Algorithm :

Step 1 : Start

Step 2 : Get the value of voltage, resistance, current and power factor

Step 3 : Compute the electric current by multiplying voltage, resistance, current and power factor with 3

Step 4 : Print the calculated electric current

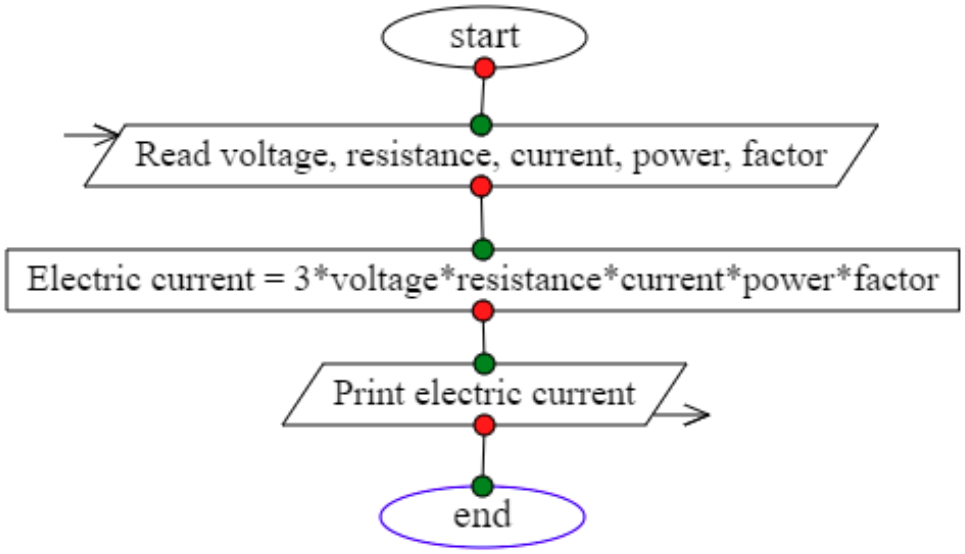
Step 5 : Stop

Pseudocode :

READ voltage, resistance, current and power factor

COMPUTE Electric current = 3 x voltage x resistance x current x power factor

PRINT Electric current



Result:

The algorithm has successfully depicted the process behind the calculation of electric current for the suitable input and desired output.

1d)

Aim: The objective of this activity is to develop a Flowchart for calculating the weight of a motorcycle.

Algorithm :

Step 1: Start

Step 2: Input weight values

Step 3: Enter the weight of the motorcycle frame.

Step 4: Enter the weight of the motorcycle engine.

Step 5: Enter the combined weight of all motorcycle wheels.

Step 6:Enter the weight of any additional accessories.

Step 7: Calculate the total weight

Step 8: Sum up the weight values to find the total weight.

Step 9: Output the total weight.

Step 10: Stop

Pseudocode :

BEGIN

TotalWeight = 0

INPUT "Enter the weight of the motorcycle frame:", FrameWeight

TotalWeight = TotalWeight + FrameWeight

INPUT "Enter the weight of the motorcycle engine:", EngineWeight

TotalWeight = TotalWeight + EngineWeight

INPUT "Enter the combined weight of all motorcycle wheels:", WheelsWeight

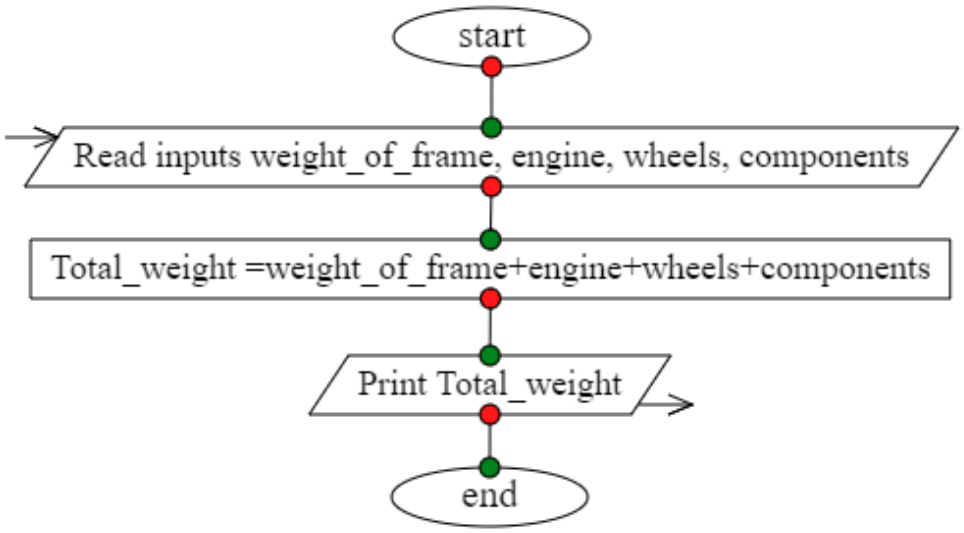
TotalWeight = TotalWeight + WheelsWeight

INPUT "Enter the weight of any additional accessories:", AccessoriesWeight

TotalWeight = TotalWeight + AccessoriesWeight

PRINT "Total weight of the motorcycle is:", TotalWeight

END



Result:

The algorithm has successfully depicted the process behind calculating the weight of a motorcycle for the suitable input and desired output.

1e)

Aim: The objective of this activity is to develop a Flowchart for generate a sine series

Algorithm :

Step 1: Start

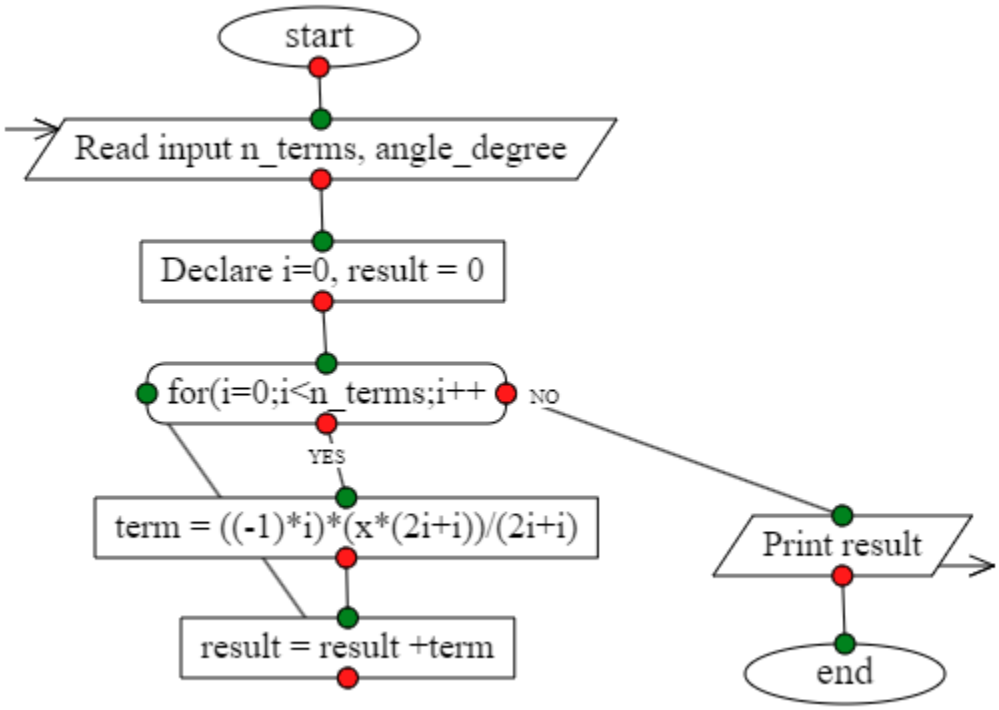
Step 2: Get Value of x.

Step 3: Using Sine series formula getting the result

Step 4: Looping the formula for each number.

Step 5: Print the result.

Step 6: Stop



Result:

The algorithm has successfully depicted the process behind calculating the sine series for the suitable input and desired output.

Ex.No.2	STATEMENTS AND EXPRESSIONS	
16.09.23		

2b)

Aim: The objective of this program is to exchange the values stored in two variables with and without using a third variable.

i). Exchanging values with third variable

Algorithm:

1. Start the program.
2. Declare three variables.
3. Assign the value of the first variable to a temporary variable.
4. Assign the value of the second variable to the first variable.
5. Assign the value of the temporary variable to the second variable.
6. End the program

Program:

```
#include<stdio.h>
int main(void) {
    // Variable declaration
    int a, b, temp;
    printf("Enter two numbers a and b ");
    scanf("%d %d", &a, &b);
    // Swap logic
    temp = a;
    a = b;
    b = temp;
```

```
        printf("\n After swapping \na = %d\nb = %d\n", a, b);
        return 0;
    }
```

Output:

```
[urk23cs1261@datalab Ex-2]$ ./a.out
Enter two numbers a and b: 12 22

Before Swapping
a=12 b=22

After Swapping
a=22 b=12
[urk23cs1261@datalab Ex-2]$
```

Result: Thus, the program executed successfully.

ii). Exchanging value of two variables without using third variable

Algorithm:

1.Start the program

2. Read the values of two variables X and Y.

3. Compute the new values of X and Y as follows,

X= X + Y.

Y= X - Y

X = X – Y

4. Print the values of X and Y

5. End the program

Program

```
#include<stdio.h>
int main() {
```

```
    int X=10, Y=20;
    printf("Before swap X=%d Y=%d",X,Y);
    X=X+Y;
    Y=X-Y;
    X=X-Y;
    printf("\nAfter swap X=%d Y=%d",X,Y);
    return 0;
}
```

Output:

```
[urk23cs1261@datalab Ex-2]
Before swap X=10 Y=20
After swap X=20 Y=10
```

Result: Thus, the program executed successfully.

2b)

Aim: The objective of this program is to find the distance between two points.

Algorithm:

Step 1: StartInput the coordinates of Point-1 as (x_1, y_1).

Step 2: Input the coordinates of Point-2 as (x_2, y_2).

Step 3: Calculate the differences in x and y coordinates

Step 4: diff_x = x_2 - x_1diff_y = y_2 – y_1

Step 5: Calculate the distance using the Pythagorean theorem

Step 6: distance = sqrt(diff_x^2 + diff_y^2)

Step 7: Output the calculated distance.

Step 8: Stop

Program:

```c
#include <stdio.h>

#include <math.h>

int main() {

   float x_1, y_1, x_2, y_2, distance, fin_x, fin_y;

   printf("Enter Point-1 (x1 y1): ");

   scanf("%f %f", &x_1, &y_1);

   printf("Enter Point-2 (x2 y2): ");

   scanf("%f %f", &x_2, &y_2);

   fin_x = x_2 - x_1;

   fin_y = y_2 - y_1;

   distance = sqrt((fin_x * fin_x) + (fin_y * fin_y));

   printf("Distance = %f\n", distance);

   return 0;

}

//gcc -o nameanything program.c -lm  → can name output file too
//cc program.c -lm → -lm for we are using math fun ,sqrt.
```

Output:

```
[urk23cs1261@datalab Ex-2]$ ./a.out
Enter Point-1 (x1 y1): 5 8
Enter Point-2 (x2 y2): 9 0
Distance = 8.944272
[urk23cs1261@datalab Ex-2]$
```

Result: Thus, the program executed successfully.

2c)

Aim: The objective of this program is to print the result of arithmetic expression.

Algorithm:

Step : 1 Start.

Step : 2 Declare integer variables a, b, and c.

Step : 3 Input two integers a and b.

Step : 4 Calculate c as the square of the sum of a and b, i.e., c = (a + b) * (a + b).

Step :5 Output the value of c.

Step :6 Stop

Program:

```
#include <stdio.h>
int main() {
   int a,b,c;
   printf("Enter two numbers a and b: ");
   scanf("%d %d",&a,&b);
   c = (a+b)*(a+b);
   printf("\nSquare of a=%d b=%d is %d\n",a,b,c);
   return 0;
}
```

Output:

```
[urk23cs1261@datalab Ex-2]$ ./a.out
Enter two numbers a and b: 12 21

Square of a=12 b=21 is 1089
```

Result: Thus, the program executed successfully.

Ex.No.3	CONDITIONALS AND ITERATIVE LOOPS	
18.09.23		

3a)

Aim: The objective of this program is to calculate the sum of n natural odd numbers.

Algorithm:

Step 1: Start the program.

Step 2: Read the range

Step 3: Use for loop to iterate up to the range

Step 4: Print the result

Step 5: Stop the program

Program:

```
#include <stdio.h>
int main() {
   int i,n,sum=0;
   printf("\nInput number of terms: ");
   scanf("%d",&n);
   printf("The odd numbers are: ");
   for(i=1;i<=n;i++){
      printf("%d ",2*i-1);
      sum+= 2*i-1;
   }
   printf("\nThe Sum odd numbers up to %d terms: %d\n",n,sum);
```

```
    return 0;

}
```

Output:

```
Input number of terms: 5
The odd numbers are: 1 3 5 7 9
The Sum odd numbers upto 5 terms: 25
[urk23cs1261@datalab Ex-3]$
```

Result: Thus, the program executed successfully.

3b)

Aim: The objective of this program is to generate patterns, right angle triangle using an asterisk

Algorithm:

Step 1: Start the program

Step 2: Read the range

Step 3: Use for loop to iterate through the rows and columns

Step 4: Print the result

Step 5: Stop the program

Program:

```
#include <stdio.h>

void main() {

   int i,j,rows;

   printf("Input number of rows : ");

   scanf("%d",&rows);

   for(i=1;i<=rows;i++) {

      for(j=1;j<=i;j++)

         printf("*");
```

```
    printf("\n");
  }
}
```

Output:

```
Input number of terms: 4
The odd numbers are: 1 3 5 7
The Sum odd numbers upto 4 terms: 16
(base) [urk23cs1261@karunya.edu@klab Ex-3]$
```

Result: Thus, the program executed successfully.

3c)

Aim: The objective of this program is to generate the sum of even number series for n terms

Algorithm:

Step 1: Start

Step 2: Declare integer variables i, n, and sum, and initialize sum to 0.

Step 3: Output "Input number of terms: " to prompt the user for input.

Step 4: Read an integer value from the user and store it in the variable 'n' using scanf.

Step 5: Output "The even numbers are: " to indicate the list of even numbers.

Step 6: Initialize a for loop with 'i' ranging from 1 to 'n'

Step 7: End of the for loop.

Step 8: Output "The Sum of even numbers up to 'n' terms: " followed by the value of 'sum'.

Step 9 : Stop

Program:

```
#include <stdio.h>

int main() {

   int i,n,sum=0;

   printf("Input number of terms: ");

   scanf("%d",&n);

   printf("The even numbers are: ");

   for(i=1;i<=n;i++){

      printf("%d ",2*i);

      sum+= 2*i;

   }

   printf("\nThe Sum even numbers up to %d terms: %d\n",n,sum);

   return 0;

}
```

Output:

```
[urk23cs1261@datalab Ex-3]$ ./a.out
Input number of terms: 5
The even numbers are: 2 4 6 8 10
The Sum even numbers upto 5 terms: 30
```

Result: Thus, the program executed successfully.

3d)

Aim: The objective of this program is to print the following number pattern

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

Algorithm:

Step 1: Start

Step 2: Prompt the user to enter the number of rows by displaying "Input number of rows: ".

Step 3: Read the user's input and store it in the 'rows' variable.

Step 4: Initialize a loop with 'i' starting from 1 and ending at 'rows' and printing the results

Step 5: Stop

Program:

```
#include <stdio.h>
int main() {
   int i,j,rows;
   printf("Input number of rows: ");
   scanf("%d",&rows);
   for(i=1;i<=rows;i++){
      for(j=1;j<=i;j++){
         printf("%d ",j);
      }
```

```
    printf("\n");
  }
  return 0;
}
```

Output:

```
[urk23cs1261@datalab Ex-3]$ ./a.out
Input number of rows: 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Result: Thus, the program executed successfully.

3e)

Aim: The objective of this program is to generate the Floyd‟s triangle using looping.

Algorithm:

Step 1: Start

Step 2: Prompt the user to enter the number of rows by displaying "Input number of rows: ".

Step 3: Read the user's input and store it in the 'rows' variable.

Step 4: Initialize a loop with 'i' starting from 1 and ending at 'rows' and making nested loops. By using newline(\n) printing the result

Step 5: Stop.

Program:

```
#include <stdio.h>
```

```
int main() {
   int i,j,x,rows;
   printf("Input number of rows: ");
   scanf("%d",&rows);
   for(i=1;i<=rows;i++){
      for(j=1;j<=i;j++){
         x+=1;
         printf("%d ",x);
      }
      printf("\n");
   }
   return 0;
}
```

Output:

```
Input number of rows: 5
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
[urk23cs1261@datalab Ex-3]$
```

Result: Thus, the program executed successfully.

Ex.No.4	FUNCTIONS	
04.10.23		

4a)

Aim: The objective of this program is to find the factorial of a number using a function.

Algorithm:

Step 1: Start the program

Step 2: In main function, declare variable n, fact and function fact1()

Step 3: Read the number n to find factorial

Step 4: call fact1(n)

Step 5: In the fact1(int n), initialize the variables x=1 and fact=1

Step 6: If x<=n then goto step 7 else goto step 12

Step 7: Calculate fact=fact * x

Step 8: Increment x by 1 and goto step 6

Step 9: Return fact

Step 10: print factorial

Step 11: Stop the program

Program:

```
#include <stdio.h>

int fact1(int);

int main() {
```

```
    int fact,n;
    printf("Enter a number to find factorial: ");
    scanf("%d",&n);
    fact = fact1(n);
    printf("The factorial of %d is: %d\n",n,fact);
    return 0;
}
int fact1(int n) {
    int x, fact=1;
    for(x=1;x<=n; x++)
    fact=fact*x;
    return fact;
}
```

Output:

```
[urk23cs1261@datalab Ex-4]$ ./a.out
Enter a number to find factorial: 4
The factorial of 4 is: 24
```

Result: Thus, the program executed successfully.

4b)

Aim: The objective of this program is to find the largest number in the list using function

Algorithm:

Step 1: Start

Step 2: Declare an integer function named function_largest that takes two parameters: an integer array input_list and an integer input_num.

Step 3: Declare integer variables input_num and an integer array input_list of size 10.

Step 4: Prompt the user for the number of elements by outputting "How many elements are present in the list: " and store the input in input_num.

Step 5: Output "What are the elements in the list?".

Step 6: Read input_num integers from the user and store them in the input_list array.

Step 7: Call the function_largest function with input_list and input_num as arguments.

Step 8: In the function_largest(input_list, input_num) function:

Step 8.1: Initialize largest_number to the first element of the input_list.

Step 8.2: Loop through the elements of input_list:

Step 8.2.1: Compare the current element with largest_number.

Step 8.2.2: If the current element is greater than largest_number, update largest_number.

Step 8.3: Output the largest_number, which is the largest number in the list.

Step 9: Stop

Program:

```
#include <stdio.h>

int function_largest(int[],int);

int input_num,input_list[10];

int main() {

   printf("How many elements are present in the list: ");

   scanf("%d",&input_num);

   printf("\nWhat are the elements in the list?\n ");

   for(int x=1; x<= input_num; x++){

      printf("\n%d ==> ",x);

      scanf("%d", &input_list[x-1]);
```

```
    }

    function_largest(input_list,input_num);

    return 0;

}

int function_largest(int input_list[], int input_num){

    int largest_number = input_list[0] ;

    for(int y=0; y<=input_num;y++){

        if(input_list[y] > largest_number){

            largest_number = input_list[y];

        }

    }

    printf("%d", largest_number);

    return 0;

}
```

Output:

```
[urk23cs1261@datalab Ex-4]$ ./a.out
How many elements are present in the list: 5

What are the elements in the list?

1 ==> 123

2 ==> 332

3 ==> 514

4 ==> 122

5 ==> 1877
Largest Number in list: 1877
```

Result: Thus, the program executed successfully.

4c)

Aim: The objective of this program is to find the area of a triangle using function.

Algorithm:

Step 1: Start the program.

Step 2: Declare the necessary variables a, b, and areaoftriangle as doubles.

Step 3: Input the user to enter the first number and read the input into a.

Step 4: Input the user to enter the second number and read the input into b.

Step 5:Calculate the area of the triangle using the formula areaoftriangle = 0.5 * a * b.

Step 6:Print the calculated area using printf with two decimal places.

Step 7: Stop

Program:

```
#include <stdio.h>

void area_of_triangle(double, double);

int main() {

   double a,b;

   printf("Enter the first number: ");

   scanf("%lf", &a);

   printf("Enter the second number: ");

   scanf("%lf", &b);

   area_of_triangle(a,b);

   return 0;

}

void area_of_triangle(double a, double b) {

   double areaoftriangle = 0.5*a*b;
```

```
    printf("\nArea of triangle: %.2lf\n", areaoftriangle);

}
```

Output:

```
[urk23cs1261@datalab Ex-4]$ ./a.out
Enter the first number: 15
Enter the second number: 20

Area of triangle: 150.00
```

Result: Thus, the program executed successfully.

Ex.No.5	STRINGS	
11.10.23		

5a)

Aim: The objective of this program is to reverse a string using a for loop.

Algorithm:

Step 1. Start the program.

Step 2. Declare a character array „str‟ of size 50 to store the input string and initialize it.

Step 3. Declare variables „i‟, „initial‟, „end‟, and „len‟ of type int.

Step 4. Print the initial given string.

Step 5. Calculate the length of the string using the „strlen()‟ function and store it in the „len‟ variable.

Step 6. Initialize „initial‟ to 0 and „end‟ to „len – 1‟.

Step 7. Use a for loop to swap characters in the string:

a. For each iteration of the loop, swap the character at index „i‟ with the character at index „end‟.

b. Increment „i‟ and decrement „end‟ to move towards the middle of the string. c. Continue this process until „i‟ is less than end.

Step 8. Print the reversed string.

Step 9. End the program.

Program:

```
#include <stdio.h>

#include <string.h>
```

```
int main(){
   char str[50]="tutorial", temp;
   int initial, end, len;
   printf(" Given String = %s \n", str);
   len = strlen(str);
   end = len - 1;
   for (initial=0; initial < end; initial++) {
      temp = str[initial];
      str[initial] = str[end];
      str[end] = temp;
      end--;
   }
   printf("\nReversed String = %s\n", str);
   return 0;
}
```

Output:

```
 Given String = tutorial

Reversed String = lairotut
[urk23cs1261@datalab Ex-5]$
```

Result: Thus, the program executed successfully.

5b)

Aim: The objective of this program is to check whether the given string is palindrome or not.

Algorithm:

Step 1. Start the program.

Step 2. Including the required pre-defined directories header files of stdio and string.

Step 3. Declaring variables and arrays word, palindrome word.

Step 4. Prompting the word from the user.

Step 5. Defining e as string length to use in for loop.

Step 6. By using string comparison, we can find the palindrome with if conditions.

Step 7. Printing the results.

Step 8 End

Program:

```
#include <stdio.h>
#include <string.h>


int main(){
   char word[20],palindrom_word[20];
   int i,e;
   printf("Want to check if your word is a palindrome or not?\nThen enter the word: ");
   scanf("%s",word);
   printf("So you want to check palindrome for this word \"%s\"!", word);
   e=strlen(word)-1;
   for(i=0; i<=e; i++){
```

```
    palindrom_word[i] = word[e-i];

  }

  if (strcmp(word,palindrom_word) == 0){

    printf("\nWord : \"%s\" is a palindrome!\n",word);

  }

  else {

    printf("\nWord : \"%s\" is not a palindrome!\n",word);

  }

  return 0;

}
```

Output:

```
[urk23cs1261@datalab Ex-5]$ ./a.out
Want to check your word is a palindrom or not?
Then enter the word: noaon
So you want to check palindrom for this word "noaon"!
Word : "noaon" is a palindrom!
```

Result: Thus, the program executed successfully.

5c)

Aim: The Objective of this program is to count the length of a string and search for a character in the string, replace a character in a string.

Algorithm:

1. Start the program.

2. Including the required pre-defined directories header files of stdio and string.

3. Declaring variables and arrays word, palindrome word.

4. Prompting the word from the user.

5. Defining e as string length to use in for loop.

6. Prompting the user again for the number where the character is present.

7. Using a for loop to check each and every letter whether they are matching or not and finally changing the word.

8. Printing the result.

9. End

Program:

```
#include <stdio.h>

#include <string.h>

int main(){

   char word[20], in_char;

   int i,e,num_letter;

   printf("Want to change a letter in word?\nThen enter the word: ");

   scanf(" %s",word);

   e=strlen(word);

    printf("\nSo now which letter you want to change in this word(len %d) :\"%s\"|\nLetter num: ",e,word);

   scanf(" %d", &num_letter);

   printf("\nFor what char you want to change this letter (%d) to ?: ",num_letter);

   scanf(" %c", &in_char);


   for (i=0; i<=e; i++){
```

```
      if (i == num_letter-1){

         word[i] = in_char;

      }

   }

   printf("Word: %s\n", word);

}
```

Output:

```
[urk23cs1261@datalab Ex-5]$ ./a.out
Want to change a letter in word?
Then enter the word: Tunvik

So now which letter you want to change in this word(len 6) :"Tunvik"|
Letter num: 2

For what char you want to change this letter (2) to ?: a
Word: Tanvik
```

Result: Thus, the program executed successfully.

Ex.No.6	VALIDATION	
18-10-23		

6a)

Aim: The Objective of this program is to create a program to validate the age of a voter.

Algorithm:

Step 1.Start

Step 2.Declare an integer variable age

Step 3.Output: "Enter your age: "

Step 4.Input the value of age

Step 5.If age is greater than or equal to 18, go to step 6, else go to step 7

Step 6.Output: "You are eligible to vote!"

Step 7.Output: "Sorry, you are not eligible to vote."

Step 8.End

Program:

```
#include <stdio.h>
int main() {
   int age;
   // Input age from user
   printf("Enter your age: ");
   scanf("%d", &age);
   // Validate age
   if (age >= 18) {
```

```
    printf("You are eligible to vote!\n");
  }
  else {
    printf("Sorry, you are not eligible to vote.\n");
  }
  return 0;
}
```

Output:

```
[urk23cs1261@datalab Ex-6]$ ./a.out
Enter your age: 19
You are eligible to vote!
[urk23cs1261@datalab Ex-6]$ ./a.out
Enter your age: 12
Sorry, you are not eligible to vote.
```

Result: Thus, the program executed successfully.

6b)

Aim: The Objective of this program is to find the marks ranges of students .

Algorithm:

Step 1. Start the program.

Step 2.Declare two variables: marks to store the student's marks and grade to store the corresponding grade.

Step 3.Prompt the user to enter their marks and read the input using scanf.

Step 4.Check the value of marks using a series of if and else if conditions:

a. If marks is equal to 100, assign "A+" to the grade using strcpy.

b. If marks are in the range of 90 to 99, assign 'A' to the first character of grade.

c. If marks are in the range of 80 to 89, assign 'B' to the first character of grade.

d. If marks are in the range of 70 to 79, assign 'C' to the first character of grade.

e. If marks are in the range of 51 to 69, assign 'D' to the first character of grade.

f. If marks are less than 50, assign 'F' to the first character of grade.

Step 5.Print the calculated grade using printf.

Step 6.End.

Program:

```
#include <stdio.h>
#include <string.h>
int main() {
   int marks;
   char grade[3];
   printf("Enter Your Marks: ");
   scanf("%d", &marks);
   if (marks == 100) {
      strcpy(grade, "A+");
   }
   else if (marks >= 90 && marks <= 99) {
      grade[0] = 'A';
   }
   else if (marks >= 80 && marks <= 89) {
      grade[0] = 'B';
   }
   else if (marks >= 70 && marks <= 79) {
      grade[0] = 'C';
   }
   else if (marks >= 51 && marks <= 69) {
      grade[0] = 'D';
```

```
    }
    else if (marks < 50) {
        grade[0] = 'F';
    }
    printf("Your Grade: %s\n", grade);
    return 0;
}
```

Output:

```
Enter Your Marks: 100
Your Grade: A+
[urk23cs1261@datalab Ex-6]$ ./a.out
Enter Your Marks: 95
Your Grade: A
[urk23cs1261@datalab Ex-6]$ ./a.out
Enter Your Marks: 85
Your Grade: B
[urk23cs1261@datalab Ex-6]$ ./a.out
Enter Your Marks: 75
Your Grade: C
[urk23cs1261@datalab Ex-6]$ ./a.out
Enter Your Marks: 65
Your Grade: D
[urk23cs1261@datalab Ex-6]$ ./a.out
Enter Your Marks: 45
Your Grade: F
```

Result: Thus, the program executed successfully.

6c)

Aim: The Objective of this program is to avoid divide by zero error.

Algorithm:

Step 1. Include header file <stdio.h>.

Step 2. Declare variables a, b, and c.

Step 3. Prompt the user to enter two numbers.

Step 4. Read the input numbers into variables a and b.

Step 5. If b is 0, display an error message and exit.

Step 6. Calculate the result c as a floating-point division of a by b.

Step 7. Display the result c with two decimal places.

Step 8. End.

Program:

```
#include <stdio.h>
int main(){
   int a,b;
   float c;
   printf("Enter two numbers: ");
   scanf("%d %d",&a,&b);
   if (b==0){
      printf("-_- Can't Divide by 0!\n");
   }
   else {
      c =(float)a/b;
      printf("Value: %.2f\n",c);
   }
   return 0;
}
```

Output:

```
[urk23cs1261@datalab Ex-6]$ ./a.out
Enter two numbers: 20 0
-_- Can't Divide by 0!
[urk23cs1261@datalab Ex-6]$
```

Result: Thus, the program executed successfully.

Ex.No.7	REAL-TIME/TECHNICAL APPLICATIONS USING FUNCTIONS	
18-10-23		

7a)

Aim: The Objective of this program is to calculate the grades for five subjects based on the marks scored using functions

Algorithm:

Step 1. Start the program.

Step 2. Declare an array to store the marks for five subjects as floating-point numbers. 3. Declare two variables to store the total marks and average marks.

Step 4. Prompt the user to enter the marks for each subject one by one using a loop, and store them in the array.

Step 5. Calculate total marks by adding all the elements of the array.

Step 6. Calculate the average mark by dividing the total by 5.

Step 7. Define a function that takes a single parameter mark and returns the corresponding grade based on the predefined grading scale.

Step 8. The function uses conditional statements to check the value of `marks` and return the appropriate grade 'A', 'B', 'C', 'D', or 'F', if the mark is >=90, 80<=, 70<=, 60<=,<60, F respectively.

Step 9. Print the grades for each subject by calling the function for each subject marks in the array, along with the subject number.

Step 10. Print the average Marks and the overall grade for the student by calling the function with the average mark.

Step 11. End the program.

Program:

```
#include <stdio.h>

char calculateGrade(float marks){
```

```
    if (marks >= 90){
        return 'A';
    }
    else if (marks >= 80){
        return 'B';
    }
    else if (marks >= 70){
        return 'C';
    }
    else if (marks >= 60){
        return 'D';
    }
    else{
        return 'F';
    }
}
int main()
{
    float subjectMarks[5],totalMarks = 0.0,averageMarks;
    printf("Enter the marks for five subjects:\n");
    for (int i = 0; i < 5; i++){
        printf("Subject %d: ", i + 1);
        scanf("%f", &subjectMarks[i]);
        totalMarks += subjectMarks[i];
    }
    averageMarks = totalMarks / 5;
    printf("\nGrades for each subject:\n");
```

```
    for (int i = 0; i < 5; i++){
        printf("Subject %d: %c\n", i + 1, calculateGrade(subjectMarks[i]));
    }
    printf("\nAverage Marks: %.2f\n", averageMarks);
    printf("Grade: %c\n", calculateGrade(averageMarks));
    return 0;
}
```

Output:

```
[urk23cs1261@datalab Ex-7]$ ./a.out
Enter the marks for five subjects:
Subject 1: 92
Subject 2: 100
Subject 3: 95
Subject 4: 99
Subject 5: 100

Grades for each subject:
Subject 1: A
Subject 2: A
Subject 3: A
Subject 4: A
Subject 5: A

Average Marks: 97.20
Grade: A
```

Result: Thus, the program executed successfully.

7b)

Aim: The Objective of this program is to compute area using functions

Algorithm:

Step 1.Include the header file <stdio.h>.

Step 2.Define a constant float PI with a value of 3.14.

Step 3.Create four functions to calculate the areas of different shapes: circle_area, rectangle_area, square_area, and triangle_area.

Step 4.Inside each area calculation function, compute the respective area using appropriate formulas and return the result.

Step 5.In the main function, declare variables for user input: choice for the shape selection and area to store the calculated area.

Step 6.Prompt the user to choose a shape and read their choice into the choice variable.

Step 7.Use a switch statement to handle different shape choices, prompting the user for the necessary parameters (e.g., radius, length, width, base, and height) and calling the appropriate area calculation function.

Step 8.Display the calculated area with two decimal places.

Step 9. End

Program:

```
#include <stdio.h>

const float PI = 3.14;

int circle_area(float radius){
   float area = PI*radius*radius;
   return area;
}

int rectangle_area(float length,float width){
   float area = length*width;
   return area;
}

int square_area(float length){
   float area = length*length;
   return area;
}

int triangle_area(float base, float height){
   float area = (1/2)*(base*height);
   return area;
```

```
}
int main(){
   int choice;
   float area;
   printf("I can find areas of these shapes :\n1.Circle\n2.Rectangle\n3.Square\n4.Triangle\nWhat shapes area you want to find? (Enter in numbers): ");
   scanf(" %d",&choice);
   switch (choice) {
   case 1:
      float radius;
      printf("\nYour choice: Circle!");
      printf("\nEnter radius of the circle: ");
      scanf(" %f",&radius);
      area = circle_area(radius);
      break;
   case 2:
      float length, width;
      printf("\nYour choice: Rectangle!");
      printf("\nEnter the value of length in the rectangle: ");
      scanf(" %f",&length);
      printf("Enter the value of width in the rectangle: ");
      scanf(" %f",&width);
      area = rectangle_area(length,width);
      break;
   case 3:
      printf("\nYour Choice: Square!");
      printf("\nEnter the value of length in the square: ");
```

```
        scanf(" %f",&length);
        area = square_area(length);
        break;
    case 4:
        float base,height;
        printf("\nYour Choice: Triangle!");
        printf("\nEnter the value of base in the triangle: ");
        scanf(" %f",&base);
        printf("\nEnter the value of height in the triangle: ");
        scanf(" %f",&height);
        area = triangle_area(base,height);
        break;
    default:
        printf("Invalid choice\n");
        return 1;
    }
    printf("\nArea = %.2f",area);
}
```

Output:

```
[urk23cs1261@datalab Ex-7]$ ./a.out
I can find areas of these shapes :
1.Circle
2.Rectangle
3.Square
4.Triangle
What shapes area you want to find? (Enter in numbers): 1

Your choice: Circle!
Enter radius of the circle: 2

Area = 12.00
```

Result: Thus, the program executed successfully.

7c)

Aim: The Objective of this program is to compute Employee Payroll using functions.

Algorithm:

Step 1. Include the header file <stdio.h>.

Step 2. Define two functions, gross_pay and tax, to calculate gross pay and tax deductions, respectively. These functions take appropriate parameters and return the calculated values.

Step 3. In the main function, declare variables to store input values and the final results: hrs_worked, hrly_pay, in_tax, fin_gross_pays, fin_taxs, and fin_amount.

Step 4. Prompt the user to enter the number of hours worked and read the input into the hrs_worked variable.

Step 5. Prompt the user to enter the hourly pay rate and read the input into the hrly_pay variable.

Step 6. Prompt the user to enter the tax percentage (in the range 1.00 to 100.00) and read the input into the in_tax variable.

Step 7. Calculate the gross pay using the gross_pay function, passing hrs_worked and hrly_pay as arguments, and store the result in the fin_gross_pays variable.

Step 8. Calculate the tax deduction using the tax function, passing fin_gross_pays and in_tax as arguments, and store the result in the fin_taxs variable.

Step 9. Calculate the net worth by subtracting fin_taxs from fin_gross_pays and store the result in the fin_amount variable. Then, display the gross pay, tax deduction, and net worth with two decimal places.

Step 10. End

Program:

```
#include <stdio.h>

float gross_pay(float hrs_worked, float hrly_pay) {
   return hrs_worked * hrly_pay;
}

float tax(float gross_pay, float in_tax) {
   return (in_tax/100) * gross_pay;
```

```
}
int main() {
   float hrs_worked, hrly_pay, fin_gross_pays, fin_taxs, fin_amount,in_tax;
   printf("Enter hours you worked: ");
   scanf("%f", &hrs_worked);
   printf("Enter hourly how much you earn: ");
   scanf("%f", &hrly_pay);
   printf("Enter tax percentage(1.00-100.00): ");
   scanf("%f", &in_tax);
   fin_gross_pays = gross_pay(hrs_worked, hrly_pay);
   fin_taxs = tax(fin_gross_pays,in_tax);
   fin_amount = fin_gross_pays - fin_taxs;
   printf("Gross Pay: ₹%.2f\n", fin_gross_pays);
   printf("Tax Deduction: ₹%.2f\n", fin_taxs);
   printf("Net worth: ₹%.2f\n", fin_amount);
   return 0;
}
```

Output:

```
[urk23cs1261@datalab Ex-7]$ ./a.out
Enter hours you worked: 10
Enter hourly how much you earn: 1000
Enter tax percentage(1.00-100.00): 2.5
Gross Pay: ₹10000.00
Tax Deduction: ₹250.00
Net worth: ₹9750.00
```

Result: Thus, the program executed successfully.

Ex.No.8	POINTERS	
25.10.23		

8a)

Aim: The objective of this program is to do arithmetic operations using pointers.

Algorithm:

Step 1.Start the program

Step 2.Read two values for doing arithmetic operations

Step 3. Assign the address of those two values to two pointers ptr1 and ptr2.

Step 4. Perform the arithmetic operations such as addition, subtraction, multiplication and division

Step 5. Print the result of each operation.

Step 6.Stop the program.

Program:

```
#include<stdio.h>
int main()
{
   int no1,no2;
   int *ptr1,*ptr2;
   int sum,sub,mult;
   float div;
   printf("Enter number1:\n");
   scanf(" %d",&no1);
   printf("Enter number2:\n");
   scanf(" %d",&no2);
   ptr1=&no1;//ptr1 stores address of no1
   ptr2=&no2;//ptr2 stores address of no2
```

```
    sum=(*ptr1) + (*ptr2);
    sub=(*ptr1) - (*ptr2);
    mult=(*ptr1) * (*ptr2);
    div=(*ptr1) / (*ptr2);
    printf("sum= %d\n",sum);
    printf("subtraction= %d\n",sub);
    printf("Multiplication= %d\n",mult);
    printf("Division= %.2f\n",div);
    return 0;
}
```

Output:

```
Enter number1:
12
Enter number2:
11
sum= 23
subtraction= 1
Multiplication= 132
Division= 1.00
(base) [urk23cs1261@karunya.edu@klab Ex-8]:
```

Result: Thus, the program executed successfully.

8b)

Aim: The objective of this program is to store n elements in an array,compute the sum of n elements and print the elements and sum using a pointer.

Algorithm:

Step 1. Start the program

Step 2. Declaring variables and arrays and prompts the user for getting the number of elements in the list .

Step 3. Read the number of values in a list.

Step 4. Assigning the size of the array but getting the number of elements in the list.

Step 5. Assign the address of the array to the pointer.

Step 6. Using FOR loop and reading the elements from the user, while taking results, take sum variable and add the input elements.

Step 7. Print the result.

Step 8.Stop the program.

Program:

```
#include <stdio.h>

int main(){
   int num_in_ls;
   printf("Enter the elements in the list: ");
   scanf("%d",&num_in_ls);
   int input_list[num_in_ls],sum;
   int* pointer = input_list;
   printf("Enter the numbers in the elements\n");
   for(int x=1; x<= num_in_ls; x++){
      printf(" %d ==> ",x);
      scanf("%d", &input_list[x-1]);
      sum += *(pointer++);
   }
   printf("%d\n",sum);
}
```

Output:

```
(base) [urk23cs1261@karunya.edu@klab Ex-8]$ ./a.out
Enter the elements in the list: 4
Enter the numbers in the elements
 1 ==> 120
 2 ==> 34
 3 ==> 22
 4 ==> 11
187
```

Result: Thus, the program executed successfully.

Ex.No.9	STRUCTURES	
2-11-23		

9a)

Aim: The objective of this program is to create a simple student information management system that allows users to store and display student details using structure

Algorithm:

Step 1. Start the program.

Step 2. Declare a structure named “Student” with the fields name, age, roll, and marks.

Step 3. Declare an integer variable “n” to store the number of students.

Step 4. Declare an array of structures named “s” to store multiple student details. Choose an appropriate size for the array (e.g., n or 100).

Step 5. Prompt the user to enter the number of students and read the input into the “n” variable.

Step 6. Use a loop to input the details of each student:

a. For each student, print " Enter Student Details:".

b. Enter the student's name and read it into the name field of the i-th element in the “s” array.

c. Enter the student's age and read it into the age field of the i-th element in the “s” array.

d. Enter the student's roll number and read it into the roll field of the i-th element in the “s” array.

e. Enter the student's marks and read it into the marks field of the i-th element in the “s” array.

Step 7. After the loop, display the student details:

a. Print “Student Details:”.

b. Use a loop to iterate through the “s” array:

i. Print the student's name, age, roll number, and marks from the “s” array.

Step 8. End the program.

Program:

```
#include <stdio.h>
struct student {
   int roll;
   char name[50];
   int age;
   float marks;
};
int main() {
   int i, n;
   printf("\nEnter the Total Number Of Student: ");
   scanf("%d", &n);
   struct student s[n];
   printf("\nEnter Student Details: \n");
   for (i = 0; i < n; i++)
   {
   printf("\nEnter Student Roll No: ");
   scanf("%d", &(s[i].roll));
   printf("Enter Student Name: ");
   scanf("%s", s[i].name);
   printf("Enter Student Age: ");
   scanf("%d", &(s[i].age));
   printf("Enter Student Marks: ");
   scanf("%f", &(s[i].marks));
   }
   printf("\n\nDisplay Student Details:\n");
   for (i = 0; i < n; i++) {
```

```
        printf("\nStudent Roll No: %d", s[i].roll);
        printf("\nStudent Name: %s", s[i].name);
        printf("\nStudent Age: %d", s[i].age);
        printf("\nStudent Marks: %f", s[i].marks);
        printf("\n\n");
       }
  return 0;
}
```

Output:

```
(base) [urk23cs1261@karunya.edu@klab

Enter the Total Number Of Student: 2

Enter Student Details:

Enter Student Roll No: 94
Enter Student Name: Burn
Enter Student Age: 21
Enter Student Marks: 90

Enter Student Roll No: 42
Enter Student Name: Knuckle
Enter Student Age: 21
Enter Student Marks: 99


Display Student Details:

Student Roll No: 94
Student Name: Burn
Student Age: 21
Student Marks: 90.000000


Student Roll No: 42
Student Name: Knuckle
Student Age: 21
Student Marks: 99.000000
```

Result: Thus, the program executed successfully.

9b)

Aim: The objective of this program is to create a simple payroll of employees using structure and pointers.

Algorithm:

Step 1. Include necessary header files.

Step 2. Define the 'employees' structure.

Step 3. Declare a global pointer to an array of 'employees' structures.

Step 4. In the 'main' function:

a. Declare a variable 'no_employees' to store the number of employees.

b. Prompt the user to input 'no_employees'.

c. Allocate memory for the 'employees' array.

d. Check for memory allocation failure.

Step 5. Input employee details in a loop and calculate relevant values.

Step 6. Display employee details in a loop.

Step 7. Free the dynamically allocated memory.

Step 8. Return 0 to indicate successful execution.

Step 9. End

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct employees {
   char name[10];
   int age;
   float hrs_worked;
   float hrly_pay;
   float tax;
```

```
    float gross_pay;
    float fin_tax;
    double fin_amount;
};
struct employees* pointer_em;
int main() {
    int no_employees = 0;
    printf("How many employees do you have: ");
    scanf("%d", &no_employees);
    pointer_em = (struct employees*)malloc(sizeof(struct employees) * no_employees);
    if (pointer_em == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }
    for (int i = 0; i < no_employees; i++) {
        printf("\nEnter the name of employee: ");
        scanf(" %30[^\n]", pointer_em[i].name);
        printf("Enter the age of %s: ", pointer_em[i].name);
        scanf(" %d", &pointer_em[i].age);
        printf("Enter the number of hours %s worked: ", pointer_em[i].name);
        scanf(" %f", &pointer_em[i].hrs_worked);
        printf("Enter the per hour pay for %s: ", pointer_em[i].name);
        scanf(" %f", &pointer_em[i].hrly_pay);
        printf("Enter the tax rate (%%) for %s: ", pointer_em[i].name);
        scanf(" %f", &pointer_em[i].tax);
        pointer_em[i].gross_pay = pointer_em[i].hrs_worked * pointer_em[i].hrly_pay;
        pointer_em[i].fin_tax = (pointer_em[i].tax / 100) * pointer_em[i].gross_pay;
```

```
        pointer_em[i].fin_amount = pointer_em[i].gross_pay - pointer_em[i].fin_tax;
    }
    for (int i = 0; i < no_employees; i++) {
        printf("\nEmployee %d: Name: %s\nAge: %d\nHours Worked: %.2f\nHourly Pay: %.2f\nGross Pay: %.2f\nTax(%%): %.2f\nTax Reduction: %.2f\nFinal Amount: %.2f\n",
            i + 1,
            pointer_em[i].name,
            pointer_em[i].age,
            pointer_em[i].hrs_worked,
            pointer_em[i].hrly_pay,
            pointer_em[i].gross_pay,
            pointer_em[i].tax,
            pointer_em[i].fin_tax,
            pointer_em[i].fin_amount
        );
    }
    free(pointer_em);
    return 0;
}
```

Output:

```
How many employees you have: 2

Enter the name of employee: Burn
Enter the age of Burn: 18
Enter the number of hours Burn worked: 15
Enter the per hour Burn pay: 1000
Enter the tax (%): 2

Enter the name of employee: Tanvik
Enter the age of Tanvik: 18
Enter the number of hours Tanvik worked: 19
Enter the per hour Tanvik pay: 2000
Enter the tax (%): 0.6

Employee 1: Name: Burn
Age: 18
Hours Worked: 15.00
Hours Pay: 1000.00
Gross Pay: 15000.00
Tax(%): 2.00
Tax Reduction: 300.00
Final Amount: 14700.00

Employee 2: Name: Tanvik
Age: 18
Hours Worked: 19.00
Hours Pay: 2000.00
Gross Pay: 38000.00
Tax(%): 0.60
Tax Reduction: 228.00
Final Amount: 37772.00
(base) [urk23cs1261@karunya.edu@klab Ex-9]$
```

Result: Thus, the program executed successfully.

Ex.No.10	FILE HANDLING	
4-11-23		

10a)

Aim: The objective of this program is to create a program which copies content of a file to another file.

Algorithm:

Step 1. Start the program

Step 2. Open the source file in read mode.

Step 3. Check if the source file was successfully opened. If not, display an

Step 4. Error message and exit the program.

Step 5. Open the destination file in write mode.

Step 6. Check if the destination file was successfully opened. If not, display an error message, close the source file, and exit the program.

Step 7. Read characters from the source file one by one until the end of the file is reached (EOF).

Step 8. Write each character to the destination file.

Step 9. lose both the source and destination files.

Step 10. Print a success message.

Step 11. Stop the program

Program:

```
#include <stdio.h>

#include <stdlib.h>

int main() {

 FILE *sourceFile, *destFile;
```

```
 char ch;
 sourceFile = fopen("source.txt", "r"); // Open the source file in read mode
 if (sourceFile == NULL) {
    perror("Error opening source file");
    return 1;
 }
 destFile = fopen("destination.txt", "w"); // Open the destination file in write mode
 if (destFile == NULL) {
    perror("Error opening destination file");
    fclose(sourceFile); // Close the source file before exiting
    return 1;
 }
 while ((ch = fgetc(sourceFile)) != EOF) {
    fputc(ch, destFile); // Write each character from source to destination  }
    printf("File copied successfully!\n");
    fclose(sourceFile);
    fclose(destFile);
    return 0;
    }
}
```

Output:

```
(base) [urk23cs1261@karunya.edu@klab Ex-10]$ ./a.out
File copied successfully!
```

Result: Thus, the program executed successfully.

10b)

Aim: Objective of this program is to find the word count and the longest word in a file.

Algorithm:

Step 1. Start the program

Step 2. Open the source file in read mode.

Step 3. Check if the words file was successfully opened. If not, display an

Step 4. Error message and exit the program.

Step 5. Increment num_words.

a. If the length of the current word is greater than largest_word_length

b. update largest_word_length.

c. copy the word to largest_word.

Step 6. Close the file.

Step 7. Print the results: num_words, largest_word_length, and largest_word.

Step 8. End.

Program:

```
#include <stdio.h>
#include <string.h>
int main(){
   char word [50]="", largest_word[50]="";
   int num_words = 0, largest_word_length=0;
   FILE *wordsfile;
   wordsfile = fopen("word.txt", "r");
   if (wordsfile == NULL){
      perror("I Can't find the file...");
   }
```

```
    else {
        while(fscanf(wordsfile, "%s",word) != EOF){
            num_words++;
            int current_words_length = strlen(word);
            if (current_words_length > largest_word_length){
                largest_word_length = current_words_length;
                strcpy(largest_word, word);
            }
        }
        printf("Words found : |%d|\nLargest Length: {%d}\nWord: [%s]\n",num_words,
largest_word_length,largest_word);
    }
    fclose(wordsfile);
    return 0;
}
```

Output:

```
(base) [urk23cs1261@karunya.edu@klab Ex-10]$ ./a.out
Words found : |10|
Largest Length: {10}
Word:  [government]
```

Result: Thus, the program executed successfully.