POKHARA ENGINNERING COLLEGE

Phirke - 8, Pokhara

(Affiliated to Pokhara University)

DEPARTMENT OF COMPUTER ENGINEERING

PROGRAMMING IN C

LABORATORY MANUAL

For

I Semester B E

Compiled By

Er Sanjish KC

Dept. of Computer Engineering

Pokhara Engineering College

Phirke - 8, Pokhara

1. Develop algorithms and flowcharts to solve various problems such as

a. **Find the largest number among three numbers.**
b. **Prime numbers**
c. **Temperature Conversion**
d. **Product of Matrices**
e. **Finding sum of the terms in series**
f. **Printing various pattern**

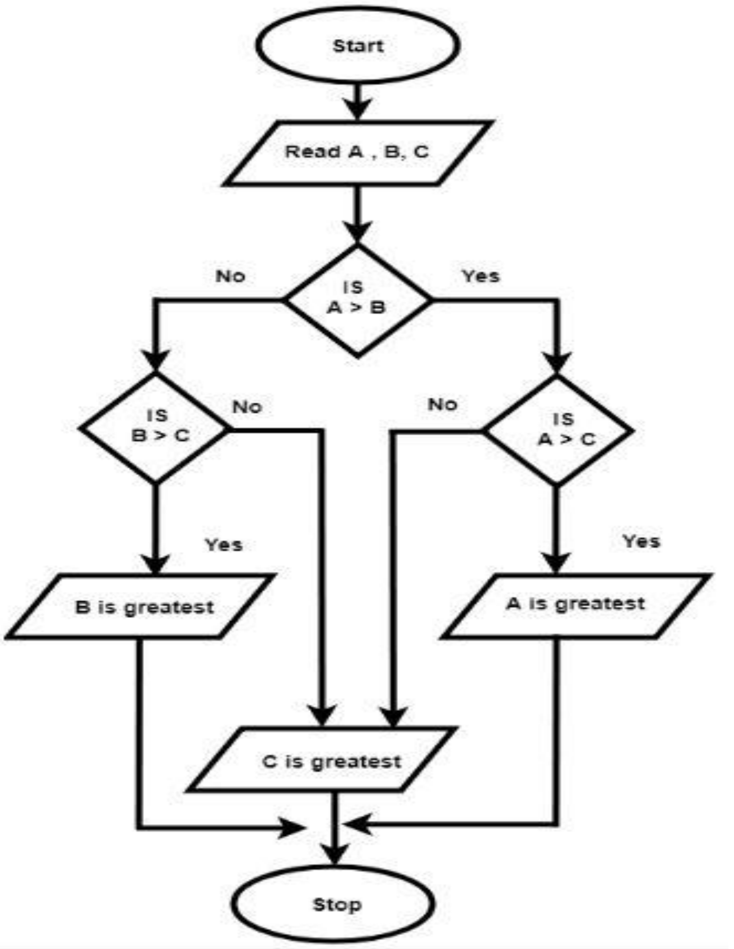
Also write C programs for the above.

Find the largest number among three numbers.

Algorithm

```
STEP 1: START
STEP 2: Read three numbers and store them in A , B, C
STEP 3: Is A > B
                If Yes: Go to Step 6,
                If No: Go to Step 4
STEP 4:  Is B > C
                If Yes: Print B is greatest
                If No: Go to Step 5
STEP 5: Print C is greatest and Go to step 8
STEP 6: Is A > C
                If Yes: Print A is greatest
                If No: Go to Step 7
STEP 7: Print C is greatest and Go to Step 8
STEP 8: Stop
```

Flowchart



Code:

```
#include <stdio.h>

int main() {
   int A, B, C;
   printf("Enter three numbers (A, B, C): ");
   scanf("%d %d %d", &A, &B, &C);

   if (A > B) {
      if (A > C) {
         printf("%d is the greatest\n", A);
      } else {
         printf("%d is the greatest\n", C);
      }
   } else {
      if (B > C) {
         printf("%d is the greatest\n", B);
      } else {
         printf("%d is the greatest\n", C);
      }
   }

   return 0;
}
```

Output 1:

```
Enter three numbers (A, B, C): 15 20 18

20 is the greatest
```

Output 2:

```
Enter three numbers (A, B, C): 55 44 88

88 is the greatest
```

Check whether the given number is prime or not.

Algorithm:

```
STEP 1: Take num as input.
STEP 2: Initialize a variable temp to 0.
STEP 3: Iterate a “for” loop from 2 to num/2.
STEP 4: If num is divisible by loop iterator, then increment temp.
STEP 5: If the temp is equal to 0,
   Return “Num IS PRIME”.
Else,
   Return “Num IS NOT PRIME”.
```

Code:

```
#include <stdio.h>
int main()
{
   int i, num, temp = 0;
   // read input from user.
   printf("Enter any numb to Check for Prime: ");
   scanf("%d", &num);
   // iterate up to n/2.
   for (i = 2; i <= num / 2; i++)
   {
      // check if num is divisible by any number.
      if (num % i == 0)
      {
         temp++;
         break;
      }
   }
   // check for the value of temp and num.
   if (temp == 0 && num != 1)
   {
      printf("%d is a Prime number", num);
   }
   else
   {
      printf("%d is not a Prime number", num);
   }
   return 0;
}
```

Output 1:

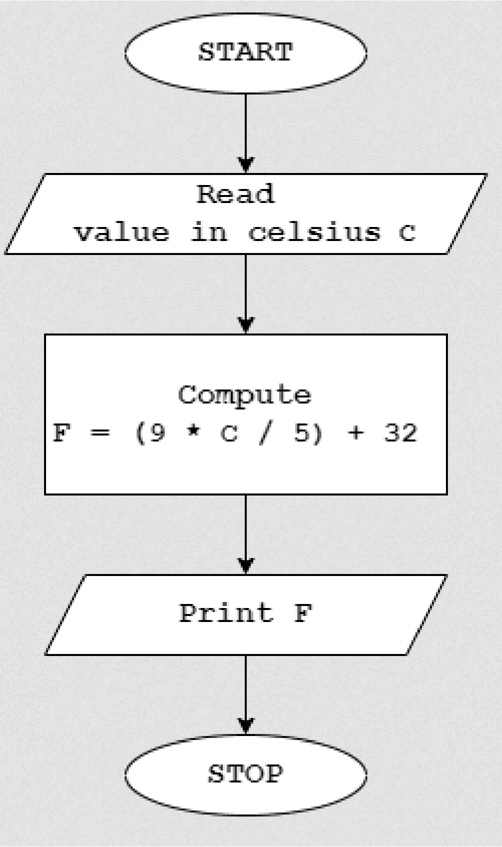
```
Enter any numb to Check for Prime: 15
15 is not a Prime number
```

Output 2:

```
Enter any numb to Check for Prime: 37
37 is a Prime number
```

Temperature Conversion

```
Step 1: Start
Step 2: Read the value of temperature to be converted from the user
Step 3: Assign the value to a variable, say ‘cel’
Step 4: Initialize f = 0
Step 5: f = ((5/9) * cel ) + 32
Step 6: Display f
Step 7: Stop
```



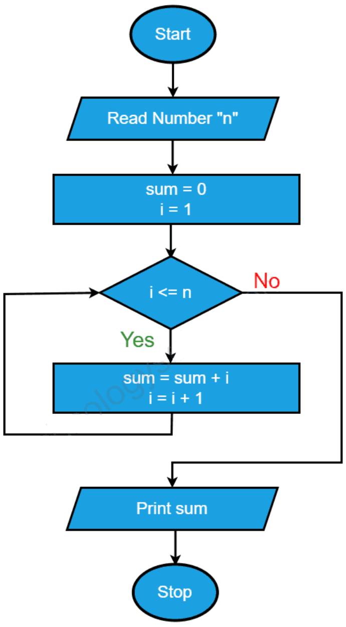
```
#include <stdio.h>

int main() {
    float cel;
    printf("Enter the temperature in Celsius: ");
    scanf("%f", &cel);

    float f = 0;
    f = (cel * 9/5) + 32;

    printf("Temperature in Fahrenheit: %.2f\n", f);

    return 0;
}
```

Finding sum of the series 1,2,3,4.....,N

Algorithm

```
Step 1: Start
Step 2: Read number n
Step 3: Declare sum to 0 and i to 1
Step 4: Repeat steps 5 to 7 until i <= n
Step 5: update sum as sum = sum + i
Step 6: increment i
Step 7: Print sum
Step 8: Stop
```

Flowchart



Code:

```
#include <stdio.h>

int main() {
    int i,n,sum=0;
    printf("Enter a number (n): ");
    scanf("%d", &n);
    for (i = 1; i <= n; i++) {
        sum = sum + i;
    }
            printf("value of i : %d\n",i);
    printf("Sum from 1 to %d is: %d\n", n, sum);
    return 0;
}
```

Output:

```
Enter a number (n): 100
Sum from 1 to 100 is: 5050
```

Printing various pattern

```
*
* *
* * *
* * * *
* * * * *
```

```
#include <stdio.h>
int main() {
   int i, j, rows;
   printf("Enter the number of rows: ");
   scanf("%d", &rows);
   for (i = 1; i <= rows; ++i) {
      for (j = 1; j <= i; ++j) {
         printf("* ");
      }
      printf("\n");
   }
   return 0;
}
```

```
* * * * *
* * * *
* * *
* *
*
```

```
#include <stdio.h>
int main() {
   int i, j, rows;
   printf("Enter the number of rows: ");
   scanf("%d", &rows);
   for (i = rows; i >= 1; --i) {
      for (j = 1; j <= i; ++j) {
         printf("* ");
      }
      printf("\n");
   }
   return 0;
}
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
#include <stdio.h>
int main() {
   int i, j, rows;
   printf("Enter the 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;
}
```

```
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
```

```
#include <stdio.h>
int main() {
   int i, space, rows, k = 0;
   printf("Enter the number of rows: ");
   scanf("%d", &rows);
   for (i = 1; i <= rows; ++i, k = 0) {
      for (space = 1; space <= rows - i;
++space) {
         printf("  ");
      }
      while (k != 2 * i - 1) {
         printf("* ");
         ++k;
      }
      printf("\n");
   }
   return 0;
}
```

```
1
2 3
4 5 6
7 8 9 10
```

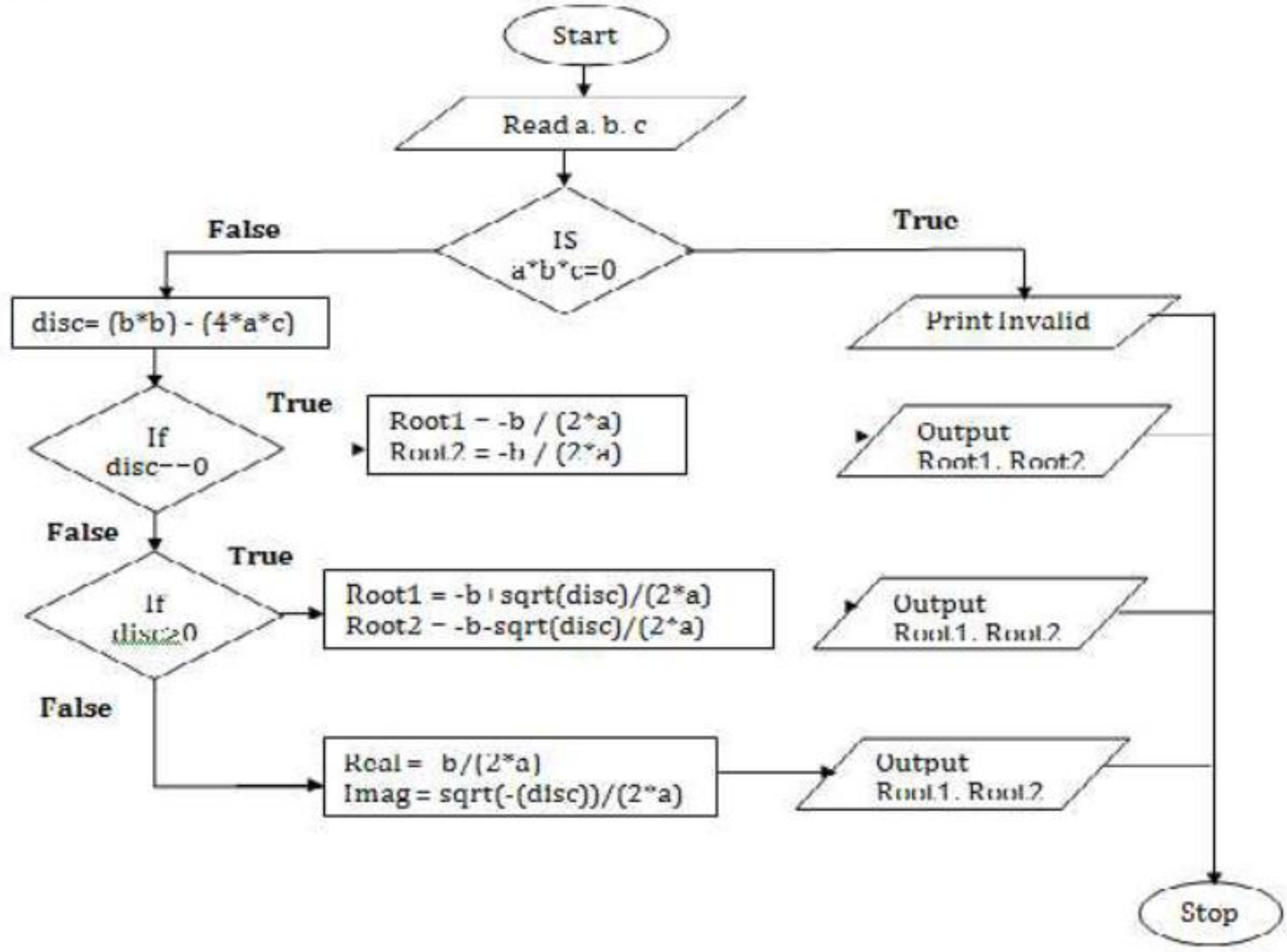
```
#include <stdio.h>
int main() {
  int rows, i, j, number = 1;
  printf("Enter the number of rows: ");
  scanf("%d", &rows);
  for (i = 1; i <= rows; i++) {
    for (j = 1; j <= i; ++j) {
      printf("%d ", number);
      ++number;
    }
    printf("\n");
  }
  return 0;
}
```

1. Design and develop a flowchart or an algorithm that takes three coefficients (a, b, and c) of a Quadratic equation (ax2+bx+c=0) as input and compute all possible roots. Implement a C program for the developed flowchart/algorithm and execute the same to output the possible roots for a given set of coefficients with appropriate messages.

Algorithm

```
Step 1. Start.
Step 2. Input co-efficient of equation a, b, c.
Step 3. IF any or all the coefficients are zero
                Print Invalid input
        ELSE
                d ← b^2- 4ac
                r ←√ |d|

        IF d > 0
                 r1 ← (-b +r)/ (2a)
                 r2 ← (-b -r)/ (2a)
                Print "Roots are REAL and DISTINCT"
                Print r1, r2
            ELSE IF d < 0
                    r1 ← -b/ (2a)
                    r2 ← r/ (2a)
                    Print "Roots are COMPLEX" Print r1 "+i" r2,
                    r1 "- i" r2
        ELSE
                    r1← -b/ (2a)
                    Print "Roots are EQUAL"
                    Print r1, r1
            END IF
            END IF
            END
            IF
```

Flowchart



```
/* C Program Find the Roots of Quadratic Equation */
#include <stdio.h>
#include <math.h>
```

```
#include <conio.h>

int main()
{
    int a, b, c;
    float d, x1, x2, r;



    printf("Enter the three coefficients:\n"); /* Accept three coefficients */
    scanf("%d%d%d", &a, &b, &c);

    if (a * b * c == 0) /* Check for zero coefficients */
    {
        printf("\n Invalid Input ");
    }
    else
    {
        d = b * b - 4 * a * c;
        r = sqrt(fabs(d));

        if (d > 0)
        {
            x1 = (-b + r) / (2.0 * a);
            x2 = (-b - r) / (2.0 * a);

            printf("\n The roots are real and distinct\n");
            printf("\n The roots are \n 1) x1=%f\t\t \n 2) x2=%f", x1, x2);
        }
        else if (d == 0)
        {
            x1 = x2 = -b / (2.0 * a);

            printf("\n The roots are real and equal\n");
            printf("\n The roots are: \n 1) x1=x2=%f", x1);
        }
        else
        {
            x1 = -b / (2.0 * a);
            x2 = r / (2.0 * a);

            printf("\n The roots are real and imaginary\n");
            printf("\n The roots are:\n 1) %f +i %f \t\t\n 2) %f -i %f", x1, x2, x1, x2);
        }
    }
            return 0;
    getch();
}
```

Sample Output:

First Run

```
Enter the three co-efficients:
1 4 4
The roots are real and equal
The roots are:
X1=X2=2.0000
```

Second Run

```
Enter the three co-efficients:
1 -5 6
The roots are real and distinct
The roots are:
X1=3.0000
X2=2.0000
```

Third Run

```
Enter the three co-efficients:
2 3 4
The roots are real and imaginary
The roots are:
1) -0.750000 +i 1.198958
2) -0.750000 - i 1.198958
```

Fourth Run

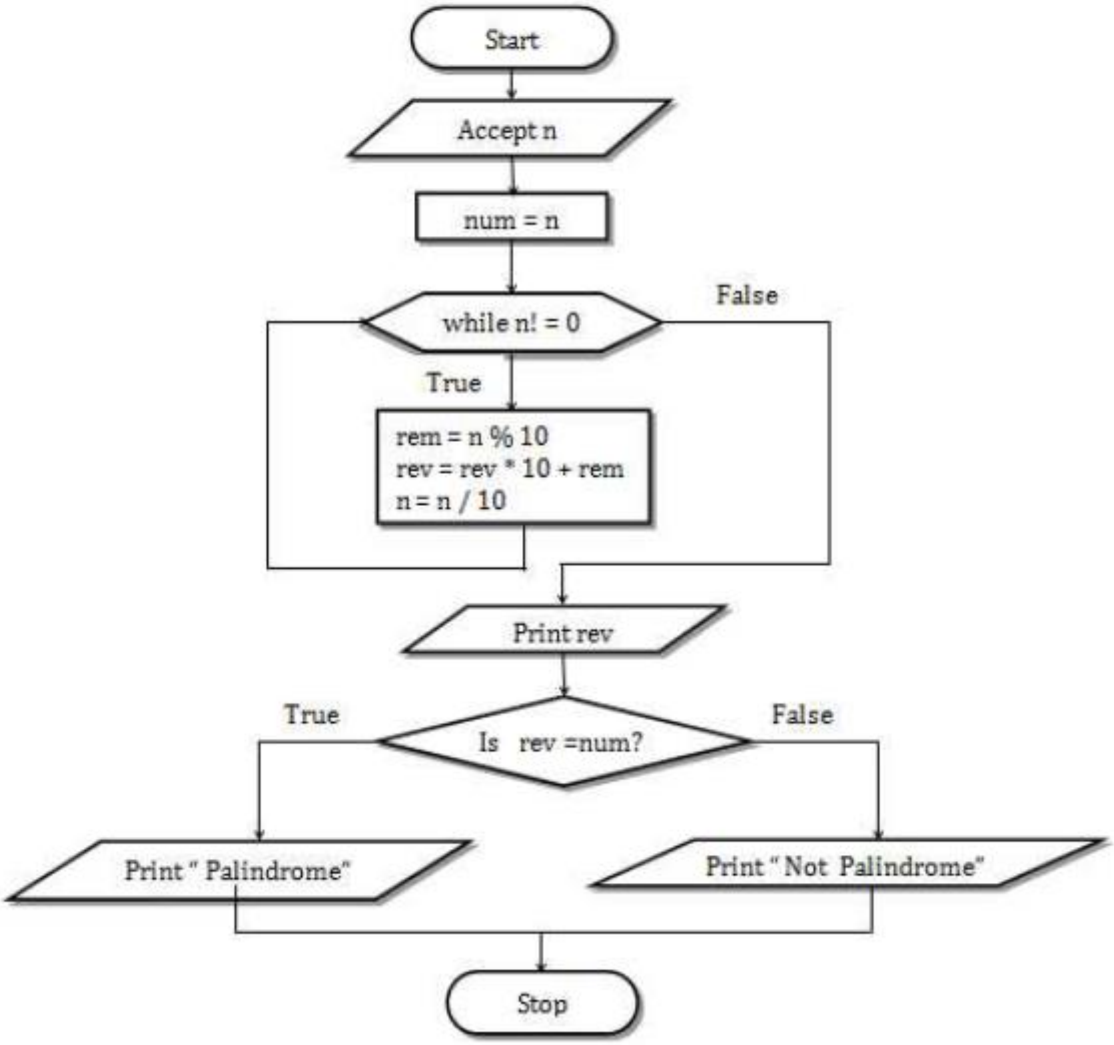
```
Enter the three co-efficients:
1 0 5
Invalid Input
```

2. Design and develop an algorithm to find the reverse of an integer number NUM and check whether it is PALINDROME or NOT. Implement a C program for the developed algorithm that takes an integer number as input and output the reverse of the same with suitable messages. Ex: Num: 2014, Reverse: 4102, Not a Palindrome.

Algorithm

```
Step 1. Start
Step 2. Input n
Step 3. Initialize num ← n, rev ← 0, rem ← 0.
Step 4.Repeat until n NOT EQUAL TO 0
            Compute rem ← n % 10
            Compute rev ← rev *10 + rem.
            Compute n ← n / 10.
      END until
Step 5. Print revs
Step 6. Check IF num EQUAL TO
            rev Print
            "Palindrome".
      ELSE
            Print "Not a Palindrome".
Step 7. Stop.
```

FLOW CHART



```
/* C Program to reverse a given integer number and check whether it is a palindrome or not. */
#include <stdio.h>

int main()
{
   int n, rev = 0, rem, num;
```

```
    printf("Enter a number: ");
    scanf("%d", &n);
    num = n;

    while (n != 0)
    {
        rem = n % 10;
        rev = rev * 10 + rem;
        n = n / 10;
    }

    printf("The reverse of %d is %d", num, rev);

    if (num == rev)
        printf("\n The given Number %d is Palindrome", num);
    else
        printf("\n The given Number %d is not Palindrome", num);

    return 0;
}
```

Sample Output

First Run

```
Enter a number:
2018
The reverse of 2018 is 8102
The Number 2018 is not Palindrome
```

Second Run

```
Enter a number:
5665
The reverse of 5665 is 5665
The Number is Palindrome
```

LAB NO: 4

Program to search an element in array.

```
#include <stdio.h>

int main() {
   int arr[] = {2, 5, 8, 12, 16, 23, 38, 45, 56, 72};
   int n = sizeof(arr) / sizeof(arr[0]);

   int key,i;
   printf("Enter the element to search: ");
   scanf("%d", &key);

   int index = -1;

   for (i = 0; i < n; i++) {
      if (arr[i] == key) {
         index = i;  // Element found, update index
         break;      // Exit the loop since the element is found
      }
   }

   if (index != -1) {
      printf("Element found at index %d\n", index);
   } else {
      printf("Element not found in the array\n");
   }

   return 0;
}
```

```
Output 1:
Enter the element to search: 23
Element found at index 5

Output 2:
Enter the element to search: 30
Element not found in the array
```

Program to find Product of Matrices

Code:

```
#include <stdio.h>

int main() {
   int m, n, p, q;

   printf("Enter dimensions for matrix A (m n): ");
```

```
    scanf("%d %d", &m, &n);

    printf("Enter dimensions for matrix B (p q): ");
    scanf("%d %d", &p, &q);

    if (n != p) {
        printf("Error: Matrix multiplication is not possible.\n");
        return 0;
    }

    int A[m][n], B[p][q], multiply[m][q];
    int i, j, k; // Declare loop counters at the beginning

    printf("Enter elements of matrix A:\n");
    for (i = 0; i < m; ++i) {
        for (j = 0; j < n; ++j) {
            scanf("%d", &A[i][j]);
        }
    }

    printf("Enter elements of matrix B:\n");
    for (i = 0; i < p; ++i) {
        for (j = 0; j < q; ++j) {
            scanf("%d", &B[i][j]);
        }
    }

    for (i = 0; i < m; ++i) {
        for (j = 0; j < q; ++j) {
            int sum = 0;
            for (k = 0; k < p; ++k) {
                sum += A[i][k] * B[k][j];
            }
            multiply[i][j] = sum;
        }
    }

    printf("Resultant matrix C:\n");
    for (i = 0; i < m; ++i) {
        for (j = 0; j < q; ++j) {
            printf("%d ", multiply[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output 1:

```
Enter dimensions for matrix A (m n): 2 3
Enter dimensions for matrix B (p q): 3 2
Enter elements of matrix A:
1 2 3
4 5 6
Enter elements of matrix B:
7 8
9 10
11 12
Resultant matrix C:
58 64
139 154
```

Output 2:

```
Enter dimensions for matrix A (m n): 2 3
Enter dimensions for matrix B (p q): 3 4
Enter elements of matrix A:
1 2 3
4 5 6
Enter elements of matrix B:
7 8 9 10
11 12 13 14
15 16 17 18
Resultant matrix C:
Error: Matrix multiplication is not possible.
```

Lab no 5: Program to pass an array to a function.

Problem Statement: Write a C program to find the sum of elements in an array using a function. Pass the array to the function and display the result in the main function.

Instructions:

1. Create a function named calculateSum that takes an array and its size as parameters and returns the sum of its elements.
2. In the main function, declare an array, initialize it with some values, and call the calculateSum function to find the sum.
3. Display the original array and the sum of its elements.

Program:

```
#include <stdio.h>

// Function to calculate the sum of elements in an array
int calculateSum(int arr[], int size) {
    int sum = 0,i;
    for (i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}

int main() {
    // Declare and initialize an array
    int myArray[] = {2, 4, 6, 8, 10};
    int i;

    // Determine the size of the array
    int size = sizeof(myArray) / sizeof(myArray[0]);

    // Calculate the sum by calling the function
    int sum = calculateSum(myArray, size);

    // Display the original array
    printf("Original Array: ");
    for ( i = 0; i < size; i++) {
        printf("%d ", myArray[i]);
    }

    // Display the sum of array elements
    printf("\nSum of Array Elements: %d\n", sum);

    return 0;
}
```

Output:

```
Original Array: 2 4 6 8 10
Sum of Array Elements: 30
```

Lab no 6: Program to use the basic string functions

Problem Statement: Write a C program that utilizes basic string functions for string manipulation. Perform the following tasks:

1. Accept a string input from the user.
2. Display the length of the entered string.
3. Display the reversed form of the entered string.
4. Copy the entered string to another string.
5. Concatenate two strings.
6. Compare two strings.

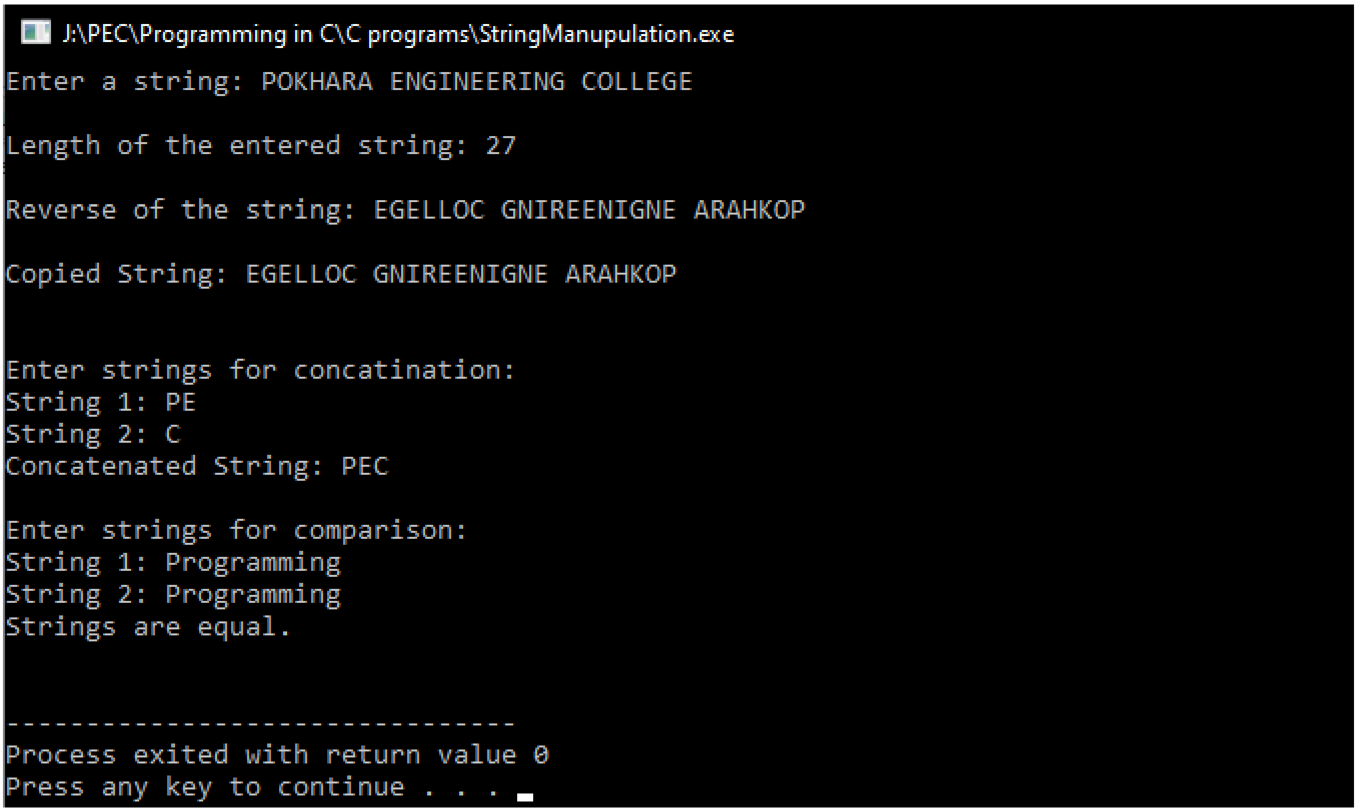
Instructions:

1. Utilize library functions such as strlen, strcpy, strcat, strcmp, and a loop for reversing the string.
2. Display the intermediate and final results clearly.
3. Ensure appropriate user prompts for input and output.

Program:

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

int main() {
    char str1[100], str2[100], copiedString[100], concatenatedString[200];
    int i;

    // Accept a string input from the user
    printf("Enter a string: ");
    scanf("%[^\n]", str1);  // Allowing spaces in the input string

    // Display the length of the entered string
    printf("\nLength of the entered string: %lu\n", strlen(str1));
        printf("\nReverse of the string: %s\n\n",strrev(str1));
    // Copy the entered string to another string
    strcpy(copiedString, str1);
    // Display the copied string
    printf("Copied String: %s\n\n", copiedString);

    // Concatenate and display two strings
    printf("\nEnter strings for concatination: ");
    printf("\nString 1: ");
    scanf("%s", str1);
    printf("String 2: ");
    scanf("%s", str2);
    strcat(str1, str2);
    printf("Concatenated String: %s\n", str1);

    // Compare two strings
    printf("\nEnter strings for comparison: ");
    printf("\nString 1: ");
    scanf("%s", str1);
    printf("String 2: ");
    scanf("%s", str2);

    int comparisonResult = strcmp(str1, str2);
    printf("Strings are %s.\n", (comparisonResult == 0) ? "equal" : "not equal");

    printf("\n");

    return 0;
}
```

Output:



J:\PEC\Programming in C\C programs\StringManupulation.exe

```
Enter a string: POKHARA ENGINEERING COLLEGE

Length of the entered string: 27

Reverse of the string: EGELLOC GNIREENIGNE ARAHKOP

Copied String: EGELLOC GNIREENIGNE ARAHKOP


Enter strings for concatination:
String 1: PE
String 2: C
Concatenated String: PEC

Enter strings for comparison:
String 1: Programming
String 2: Programming
Strings are equal.


--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Lab no 7: Program to solve the problem using recursion.

a. Factorial of a number

```
#include <stdio.h>

long int fact(int n);
int main()
{
    int n;
    printf("Enter a positive number: ");
    scanf("%d", &n);
    printf("Factorial of %d = %ld", n, fact(n));
    return 0;
}
long int fact(int n)
{
    if (n > 1)
        return n*fact(n-1);
    else
        return 1;
}
```

Output:

```
Enter a positive number: 5
Factorial of 5 = 120
```

b. Fibonacci series.

```
#include<stdio.h>
int Fibonacci(int);
int main()
{
   int n, i;
   printf("Enter the term:");
   scanf("%d",&n);
   printf("Fibonacci series:\n");
   for ( i = 0 ; i < n ; i++ )
   {
      printf("%d\n", Fibonacci(i));
   }
  return 0;
}
 int Fibonacci(int n)
{
   if ( n == 0 )
      return 0;
   else if ( n == 1 )
      return 1;
   else
      return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

Output:

```
Enter the term:10
Fibonacci series:
0
1
1
2
3
5
8
13
21
34
```