# 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**

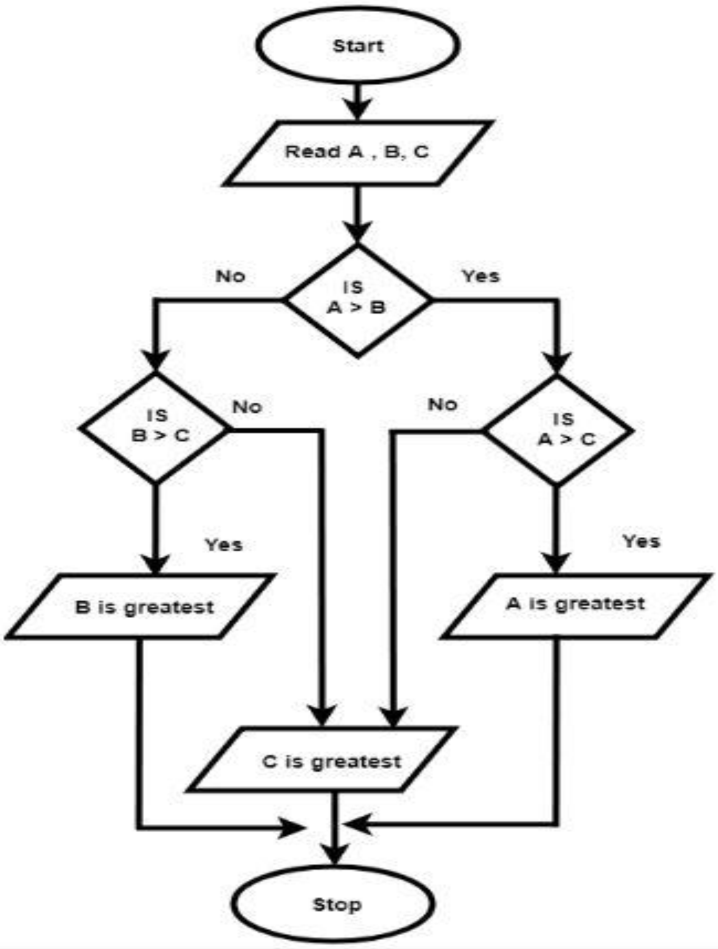
**Also write C programs for the above.**

## 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:**

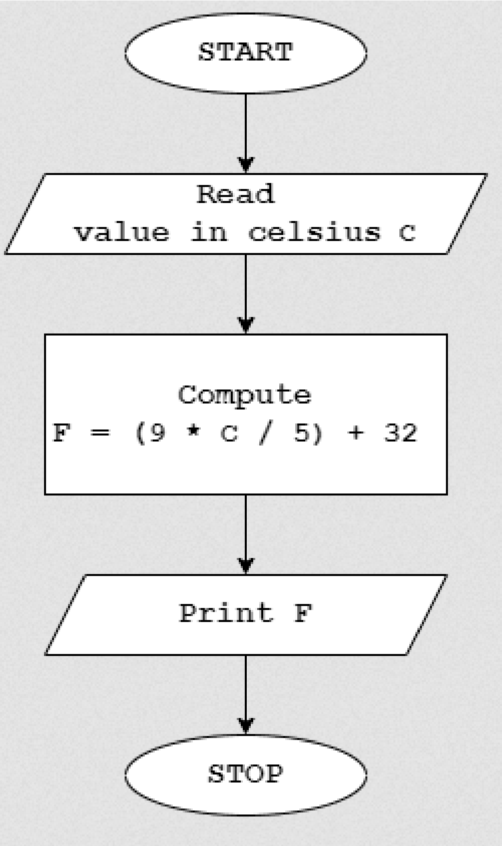
```
Enter any numb to Check for Prime: 15
15 is not a Prime number
```

**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



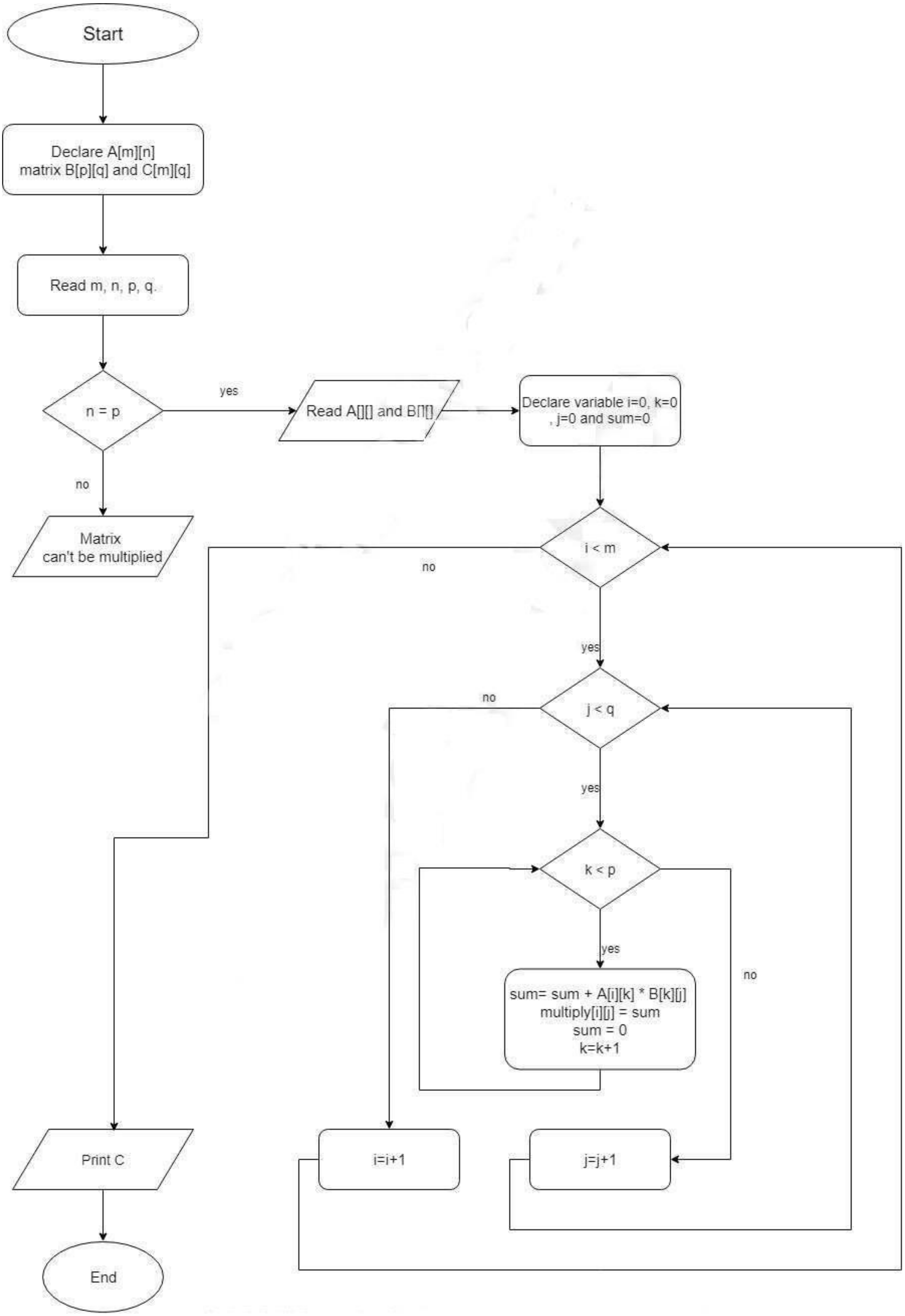
```
#include <stdio.h>

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;
}
```

# Product of Matrices

**Algorithm:**

```
Step 1: Start
Step 2: Declare matrix A[m][n]
                    and matrix B[p][q]
                    and matrix C[m][q]
Step 3: Read m, n, p, q.
Step 4: Now check if the matrix can be multiplied or not, if n is not equal to q matrix can't be multiplied and an error message is generated.
Step 5: Read A[][] and B[][]
Step 4: Declare variable i=0, k=0 , j=0 and sum=0
Step 5: Repeat Step until i < m
    5.1: Repeat Step until j < q
        5.1.1: Repeat Step until k < p
                    Set sum= sum + A[i][k] * B[k][j]
                    Set multiply[i][j] = sum;
                    Set sum = 0 and k=k+1
                    5.1.2: Set j=j+1
             5.2: Set i=i+1
Step 6: C is the required matrix.
Step 7: Stop
```

**Flowchart:**


Start
Declare A[m][n]
matrix B[p][q] and C[m][q]
Read m, n, p, q.
n = p
yes
Read A[][] and B[][]
Declare variable i=0, k=0
, j=0 and sum=0
no
Matrix
can't be multiplied
i < m
no
yes
j < q
no
yes
k < p
yes
no
sum= sum + A[i][k] * B[k][j]
multiply[i][j] = sum
sum = 0
k=k+1
Print C
i=i+1
j=j+1
End

**Code:**

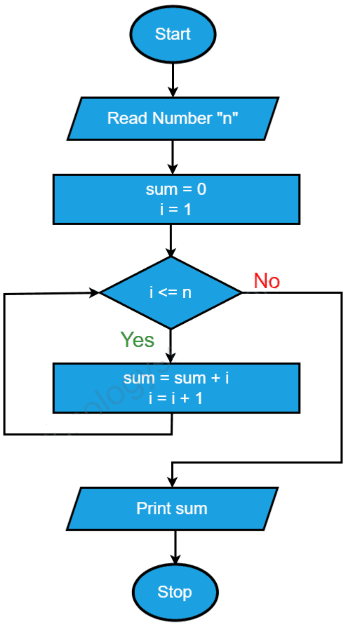
```
#include <stdio.h>

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];

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

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

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

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

    return 0;
}
```

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

### Algorithm

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

### Flowchart



### Code:

```
#include <stdio.h>

int main() {
    int n,sum=0;
    printf("Enter a number (n): ");
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        sum = sum + 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;
}
```

```
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, 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;
}
```

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

```
#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
```

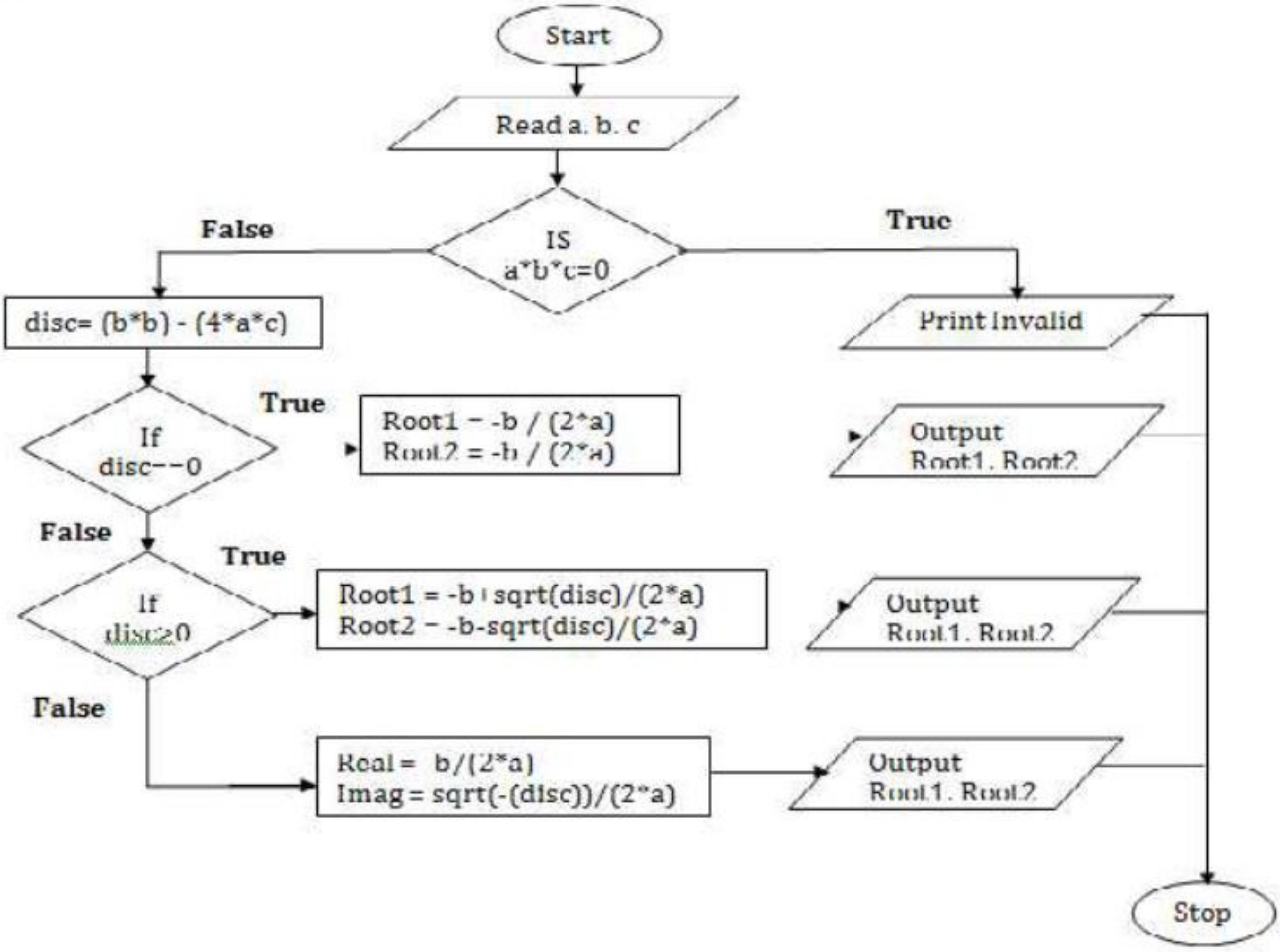
```
#include <stdio.h>
int main() {
  int rows, i, j, number = 1;
  printf("Enter the number of rows: ");
  scanf("%d", &rows);
  for (i = 1; i <= rows; i++) {
    for (j = 1; j <= i; ++j) {
      printf("%d ", number);
      ++number;
    }
    printf("\n");
  }
  return 0;
}
```

**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.
Step 4. Stop
```

## Flowchart



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

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

   clrscr();

   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);
      }
   }

   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
```

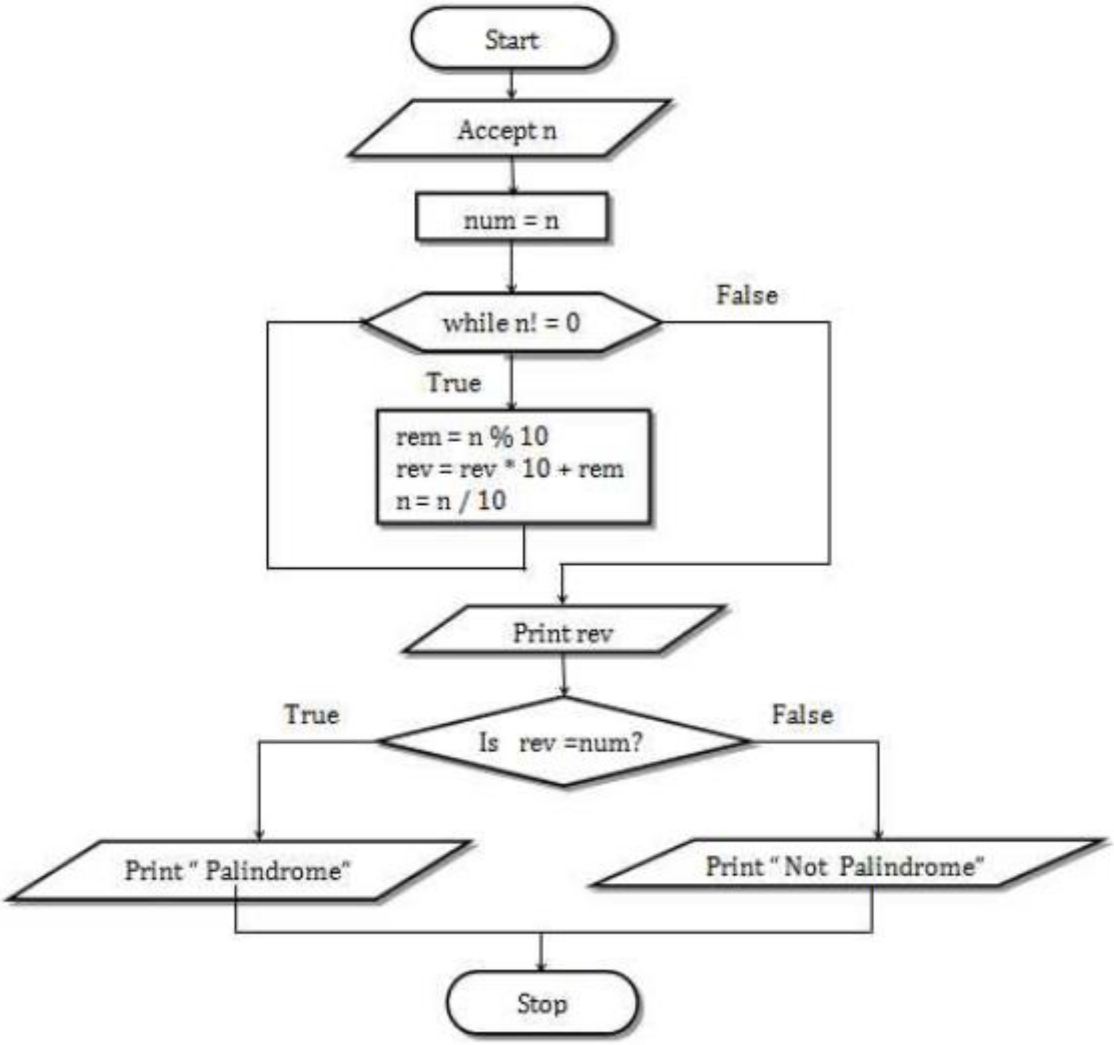
**2. Design and develop an algorithm to find the reverse of an integer number NUM and check whether it is PALINDROME or NOT. Implement a C program for the developed algorithm that takes an integer number as input and output the reverse of the same with suitable messages. Ex: Num: 2014, Reverse: 4102,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