Write a program, an algorithm, flow chart to print even numbers from 2 to 100.

Algorithm:

1. START
2. n ← 2
3. Print "Even no's from 2 to 50: \n";
4. If n<=50 goto (5) else goto (8)
5. Print n
6. n ← n+2
7. goto (4)
8. STOP

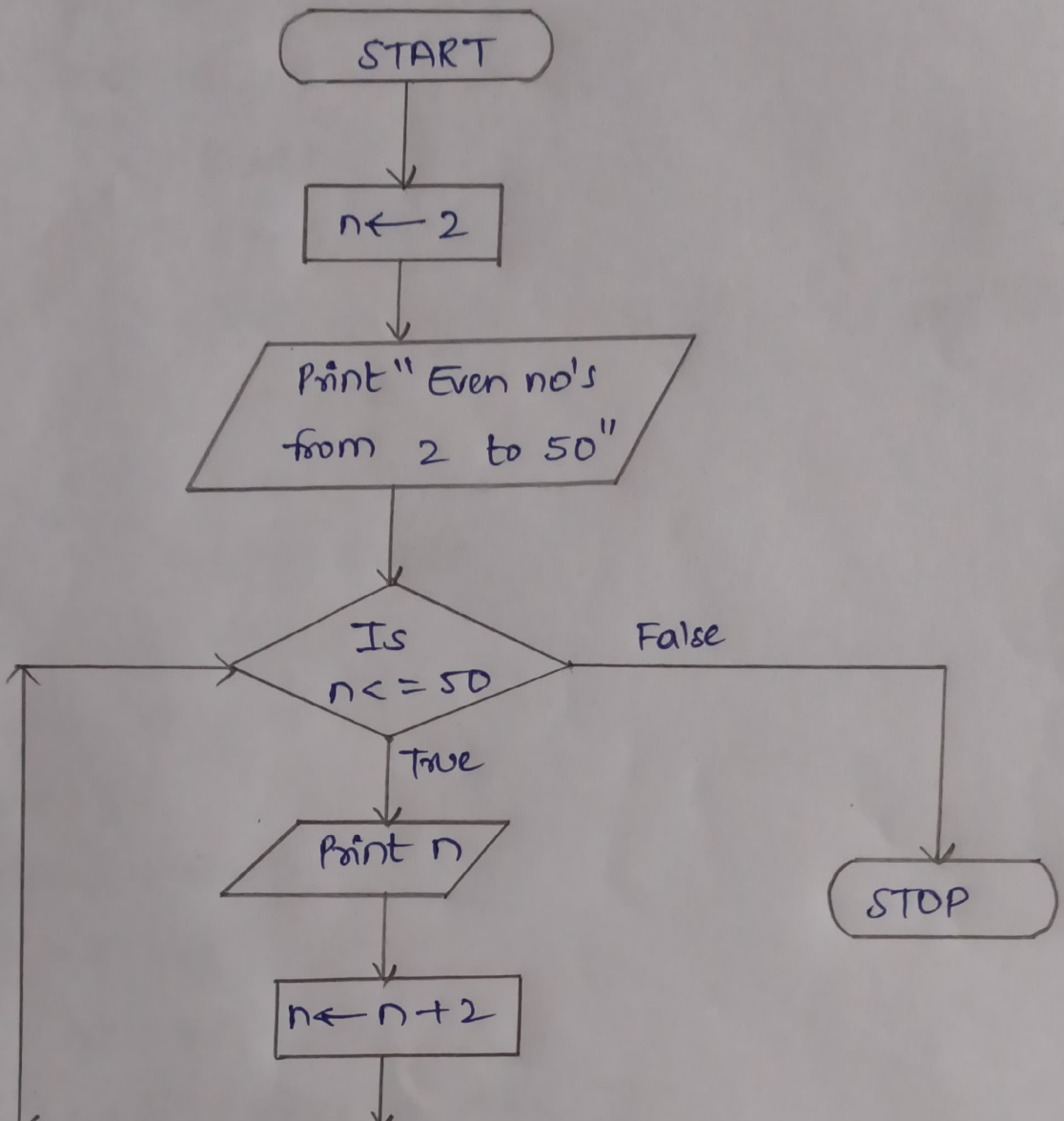
Program:

```
#include<stdio.h>
void main(){
    int n=2;
    printf("Even no's from 2 to 50:\n");
    while(n<=50){
        printf("%d ", n);
        n+=2;
    }
}
```

Output: Even no's from 2 to 50:

2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
32 34 36 38 40 42 44 46 48 50

Flow chart:


START
n ← 2
Print " Even no's from 2 to 50"
Is n <= 50
False
True
Print n
n ← n+2
STOP

Write an algorithm, program and flowchart to read a number n & find no.of digits it has.

Algorithm: 1. START

2. Read n

3. c ← 0

4. if(n! =0) goto step (5) else goto (8)

5. n ← n/10

6. c ← c+1

7. goto step (4)

9. Print c

10. STOP

Program:

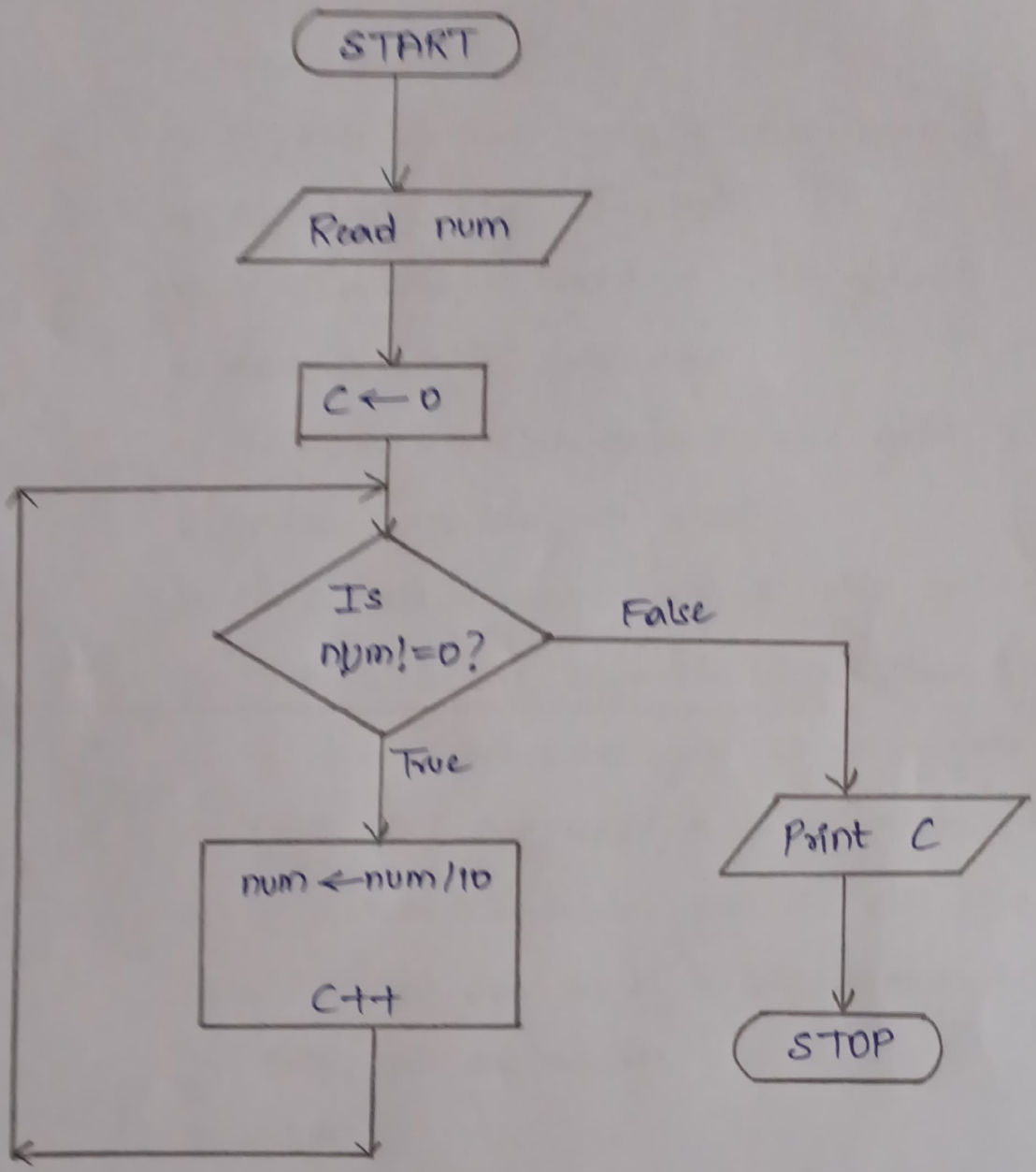
```
#include<stdio.h>
void main(){
    int n,c;
    printf("Enter a num: ");
    scanf("%d",&n);
    for(c=0; n!=0; c++)
        n/=10;
    printf("\n No.of digits: %d ", c);
}
```

Output:

Enter a num: 9654821

No.of digits : 7

FLOWCHART:



Write an algorithm, program and flowchart to find biggest of three number.

Algorithm:

1. START
2. Read a, b & C
3. if a>B && a>C goto 4 else goto 5
4. Print A is big ⇒ STOP
5. if B>a && B>C goto 6 else goto 7
6. Print B is big ⇒ STOP
7. if C>a && C>b goto 8 else goto 9
8. Print C is big ⇒ STOP
9. if a==B && a>C goto 10 else goto 11
10. Print a, B are equal & bigger than C ⇒ STOP
11. if B==C && b>a goto 12 else goto 13
12. Print B, C are equal & bigger than a ⇒ STOP
13. if C==a && a>b goto 14 else goto 15
14. Print C, a are equal & bigger than b ⇒ STOP
15. Print all are equal
16. STOP

Program:

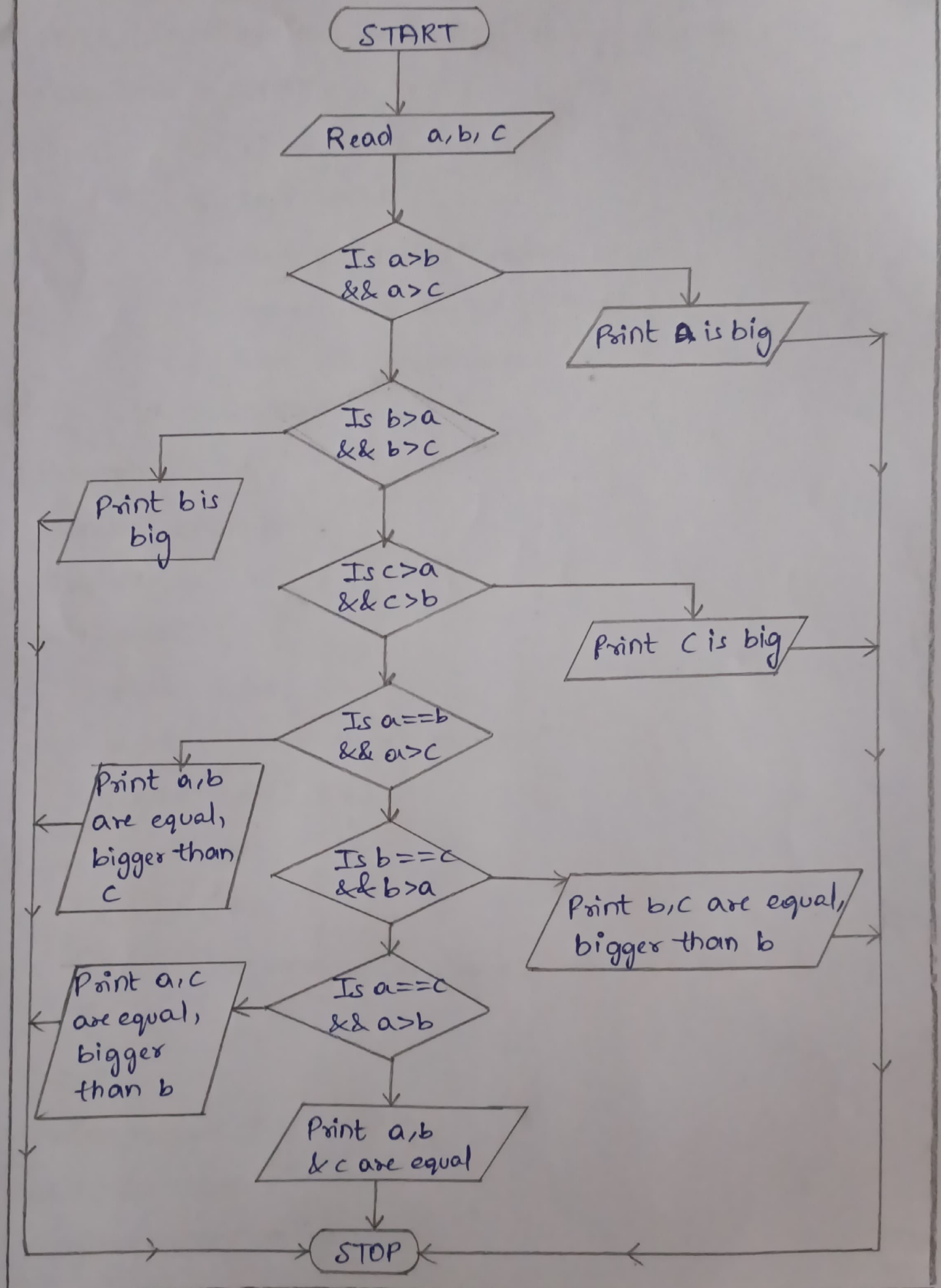
```
#include <stdio.h>
int main(){
   int a,b,C ;
   printf(" enter three nums: ");
```

```
    scanf("%d %d %d",&a,&b,&c);
    if(a>b&&a>c)
        printf("a is big");
    else if(b>a&&b>c)
        printf("b is big");
    else if(c>a&&c>b)
        printf("c is big");
    else if(a==b&&a>c)
        printf("\n a,b are equal & bigger than c");
    else if(b==c&&b>a)
        printf("\n b,c are equal & bigger than a");
    else if(a==c&&a>b)
        printf("\n a,c are equal & bigger than b");
    else
        printf("a,b,c are equal");
    return 0;
}
```

Output: Enter three nums: 10 20 20

b,c are equal & bigger than a.

FLOWCHART:


START
Read a, b, c
Is a>b && a>c
Print A is big
Is b>a && b>c
Print b is big
Is c>a && c>b
Print C is big
Is a==b && a>c
Print a,b are equal, bigger than c
Is b==c && b>a
Print b,C are equal, bigger than b
Is a==c && a>b
Print a,C are equal, bigger than b
Print a,b & c are equal
STOP

Write a program to calculate area, circumference of a circle.

Write algorithm, flow chart to the above program.

Algorithm: 1. START

2. pi ← 3.14

3. Read radius

4. A ← pi * radius * radius

5. Circumference ← 2 * pi * radius

6. Print A, circumference

7. STOP

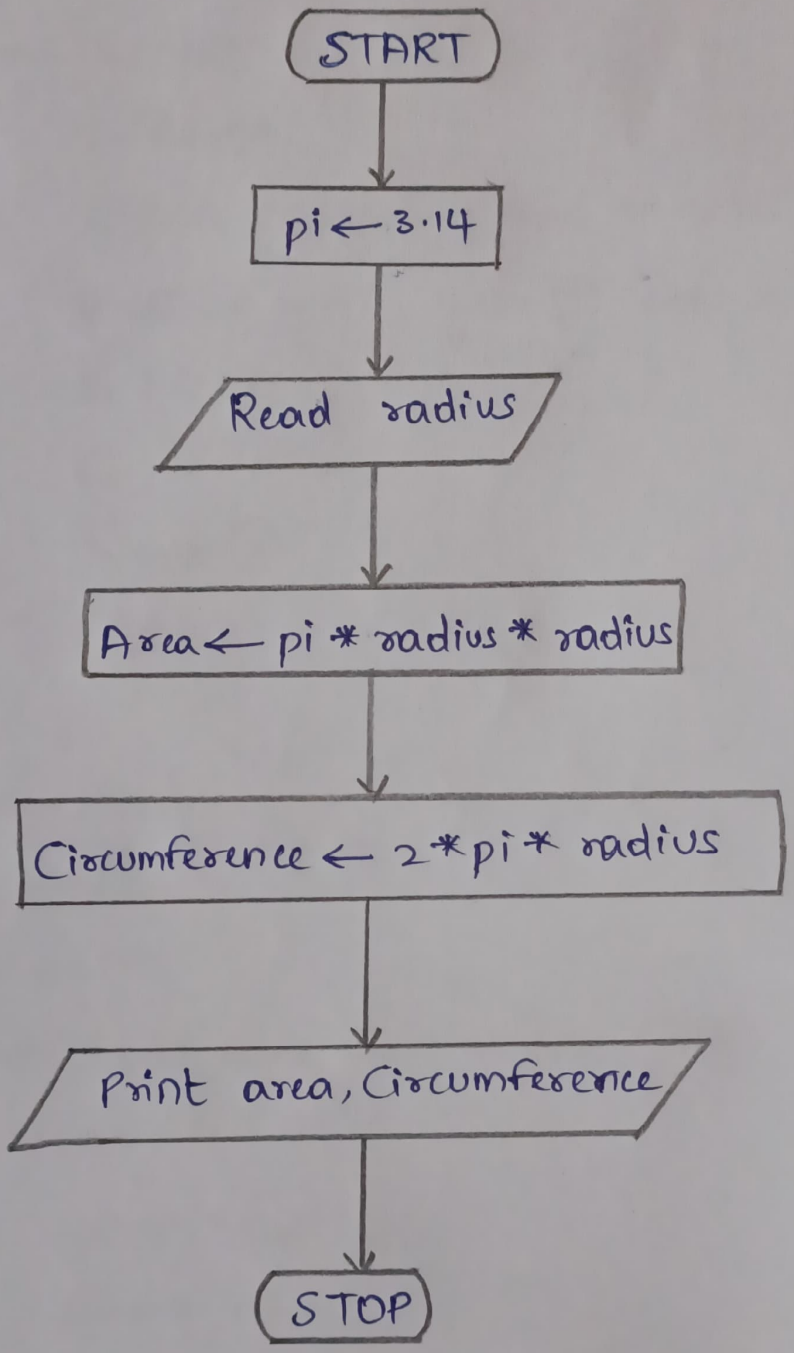
Program:

```
#include <stdio.h>
#define pi 3.14
void main(){
    float a,c,r;
    printf("Enter circle's radius: ");
    scanf("%df",&r);
    a= pi*r*r;
    c=2*pi*r;
    printf("\n area : %f \n circumference: %f ",a,c);
}
```

Output:

```
Enter radius: 5.3
area : 88.202606
 circumference: 33.284000
```

Flowchart:


START
pi ← 3.14
Read radius
Area ← pi * radius * radius
Circumference ← 2 * pi * radius
Print area, Circumference
STOP

Write an algorithm, flowchart and program to print odd numbers from 99 to 1.

Algorithm:

1. START
2. n ← 99
3. Print "Odd nums from 99 to 71: \n"
4. If n >= 71 goto (5) else goto (8)
5. Print n
6. n ← n-2
7. goto (4)
8. STOP

Program:

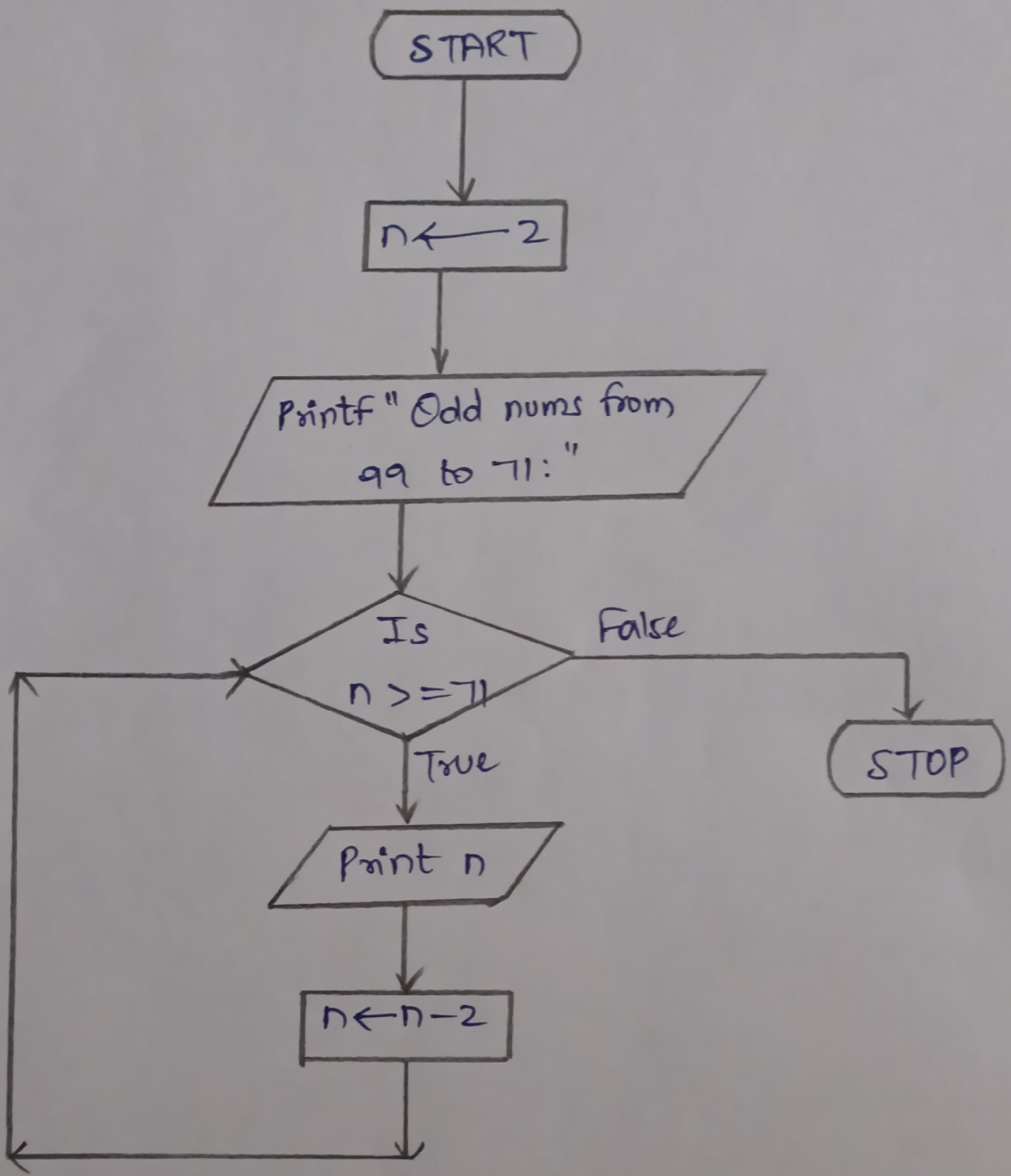
```
#include<stdio.h>
void main(){
    int n=99;
    printf("odd nums from 99 to71: \n");
    while(n>=71){
        printf("%d ",n);
        n-=2;
    }
}
```

Output: Odd nums from 99 to71:

99 97 95 93 91 89 87 85 83 81 79 77

75 73 71

Flow chart:


START
n ← 2
Printf " Odd nums from 99 to 71: "
Is n >= 71
False
True
STOP
Print n
n ← n-2

Write a program to implement linear search.

```
#include <stdio.h>
void main(){
int s, arr[100],i,n,ch;
printf("size of the array is: \n");
scanf("%d",&n);
printf("Enter %d elements : ", n);
for(i=0; i<n,i++)
    scanf("%d", &arr[i]);
do{
printf("Enter search element: \n");
scanf("%d",&s);
for(i=0; i<n,i++){
    if(arr[i]==s)
        break;
}
if(i<n)
    printf("key found at index: %d \n", i);
else
    printf("key not found \n");
printf("Enter 1 to search one more element \n");
printf("Enter 0 to exit \n");
```

```
    scanf("%d", &ch);
  } while (ch == 1);
}
```

Output:

```
Size of the array is: 5
Enter 5 elements : 5 10 15 25 20
Enter search element :
25
key found at index:3
Enter 1 to search one more element.
Enter 0 to exit
0
```

Write a program for binary search.

```
#include <stdio.h>
int main(){
  int s,arr[100],i,n,l,h, m,ch;
  printf("Enter size of the array : \n");
  scanf("%d",&n);
  printf("Enter list in ASCENDING ORDER \n");
  for(i=0; i<n; i++){
      scanf("%d" &arr[i]); }
  do{
     printf(" Enter search element: \n");
     scanf("%d" &s);
     l=0; h=n-1;
     while(l<=h){
          m=(l+h)/2;
          if (s==arr[m])
               break;
          elseif(s<arr[m])
               h=m-1;
          else
               l=m+1;
     }
     if(l<=h)
        printf(" Element found at index : %d", m);
```

```
    else
        printf("Element not found: \n");
    printf(" Enter 1 to search more & 0 to exit \n");
    scanf("%d", &ch);
  } while (ch==1);
    return 0;
}
```

Output:

```
Enter size of the array:
3
Enter list in ASCENDING ORDER:
3 6 9
Enter search element: 7
Element not found.
Enter 1 to search more & 0 to exit
1
Enter size of the array:
Enter search element
6
Element found at index :1
```

Write a program to implement insertion sort.

```
#include<stdio.h>
void main(){
    int list[100], n, i, j, k, x, temp;
    printf("Enter no.of elements to sort: ");
    scanf("%d", &n);
    printf("Enter elements one by one: \n");
    for(i=0; i<n; i++)
    {
        scanf("%d", &x);
        k=i-1;
        while(k>=0 && x<list[k]){
            list[k+1] = list[k];
            k--;
        }
        list[k+1] = x;
    }
    printf("Sorted list : \n");
    for(i=0; i<n; i++)
        printf("%d \t", list[i]);
}
```

Output: Enter no.of elements to sort: 5

Enter elements one by one

10

13

12

15

2

Sorted list:

2 10 12 13 15

Write program to sort elements using Quicksort technique.

```
#include <stdio.h>
# include <stdlib.h>
void swap(int *x, int *y){
int temp;
temp = *x;
*x = *y;
*y = temp;
}

void Qsort(int A[], int l, int h){
    if(l<h){
        int i=l, j=h;
        int p=A[l];
        while(i<=j){
            while(A[i]<p]
                i++;
            while(A[j]>p)
                j--;
            if(i<=j){
                swap(&A[i], &A[j]);
                i++;
                j--;
            }
    }
```

```
    Qsort(A,l,j);
    Qsort(A,i,h);
   }
}

void main(){
    int *list, n,i;
    printf("How many elements to sort: ");
    scanf("%d" &n);
    list =(int *)malloc(n *sizeof(int));
    printf("Enter %d elements:\n", n);
    for(i=0;i<n;i++){
        scanf("%d", &list[i]); }
    Qsort(list,0,n-1);
    printf("After Quick sort :\n");
    for(i=0; i<n;i++)
        printf("%d \t", list[i]);
    free(list);
}
```

Output: How many elements to sort: 5

Enter 5 elements:

5 3 10 20 12

After Quick sort:

3 5 10 12 20

Write a program to sort elements using merge sort.

```
#include<stdio.h>
#include<stdlib.h>
void msort(int list[], int l, int m, int h){
int A[100], k=0;
int i=l, j=m+1;
while(i<=m && j<=h){
    if (list[i]<list[j])
        A[k++] = list[i++]
    else
        A[k++] = list[j++]
}
while (i<=m)
    A[k++]=list[i++];
while(j<=h)
    A[k++] = list[j++];
for(i=0; i<k; i++)
    list[l+i] = A[i];        }
void split (int list[], int l, int h){
  if(l<h){
    int m=(l+h)/2;
    split(list, l,m);
    split(list, m+1,h);
```

```
        msort( list, l, m, h);
    }
}

void main(){
    int *A, n, i;
    printf(" How many elements to sort: ");
    scanf("%d" &n);
    A= (int *) malloc(n*sizeof(int));
    printf(" Enter %d elements: \n", n);
    for (i=0; i<n; i++)
        scanf( "%d", &A[i]);
    split (A, 0, n-1);
    printf(" Elements after Merge sort: \n");
    for(i=0, i<n; i++)
        printf("%d \t", A[i] );
    free(A);
}
```

Output: How many elements to sort: 5

Enter 5 elements:

5 3 10 20 12

Elements after merge sort:

3 5 10 12 20

Write a program to demonstrate selection sort.

```
#include<stdio.h>
void main(){
    int a[50], n,i, j, temp;
    printf(" Enter no.of elements to sort: ");
    scanf("%d",&n);
    printf(" Enter %d elements : \n",n);
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    for(i=0; i<n-1; i++){
    for(j=0; j=i+1; j<n; j++){
    if(a[i]>a[j]){
        temp =a[i];
        a[i] = a[j];
        a[j] =temp;
  }}}
  printf("Sorted list: \n");
  for(i=0; i<n; i++)
      printf("%d\t", a[i]);
}
```

Output: Enter no.of elements to sort: 5

Enter 5 elements :

5 42 17 30 45

Sorted list : 5 17 30 42 45

Write a program to demonstrate Bubble sort.

```
#include<stdio.h>
int main(){
    int a[50], n,i,j,temp;
    printf("Enter no.of elements to sort:");
    scanf("%d" &n);
    printf("Enter %d elements: \n", n);
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    for(i=1; i<n-1; i++){
    .   for(j=0; j<n-i-1; j++){
            if(a[j]>a[j+1]){
                temp = a[j];
                a[j]= a[j+1];
                a[j+1] = temp;
            }
    }}
    printf("Sorted list: ");
    for(i=0; i<n; i++)
        printf("%d \t", a[i]);
  return 0; }
```

Output: Enter no.of elements to sort: 5

Enter %d5 elements: 5 70 12 3 25

Sorted elements: 3 5 12 25 70

Write a program to find prime numbers in given range

```
#include<stdio.h>
int isPrime(int x){
    int i;
    if (x<=1) return 0;
    for(i=2; i<=x/2; i++){
        if (x%i==0)
            return 0;
        return 1;
}}
void main(){
    int start, end, n;
    printf("Enter the range start & end values(start<end): ");
    scanf("%d %d",&start,&end);
    printf("Prime no's blw %d & %d are: \n", start, end);
    for(n= start ; n<=end; n++){
        if(isPrime(n)){
            printf("%d, ",n);
    }}  printf("\n");
}
```

Output:

Enter the range start & end values(start < end): 67 150

Prime no's blw 67 & 150 are: 67, 71, 73, 79, 83, 89, 97

101, 103, 107, 109, 113, 127, 131, 139, 149,

Write a program for area & perimeter of rectangle using single function and pointers.

```
#include <stdio.h>
void Rect(int l, int b, int *A, int *P){
*A = l * b;
*P = 2*(l+b);
}
Void main(){
    int l, b, ar, pr;
    printf("Enter length & breadth of rectangle: \n");
    scanf("%d%d", &l, &b);
    Rect(l, b, &ar, &pr);
    printf("Area is: %d \n Perimeter is: %d \n", ar, pr);
}
```

Output:

```
Enter length & breadth of rectangle: 9 7
Area is 63
Perimeter is 32
```

Write a program to print fibonacci series using recursive fn.

```
#include <stdio.h>
int fib(int n){
    if(n<=1)
        return n;          //fib(0)=0 ; fib(1)=1
    return (fib(n-1)+fib(n-2));
}
int main(){
    int n;
    printf(" Enter no.of terms : ");
    scanf("%d", &n);
    printf(" Fibonacci series : ");
    for(int i=0; i<n; i++){
        printf("%d ", fib(i));
    }
    printf("\n");
    return 0;
}
```

Output:

```
Enter no.of terms: 5
Fibonacci series : 0 1 1 2 3
```

Write a program two find GCD of 4 nos using recursion.

```
#include<stdio.h>
int GCD(int m, int n){
    if(m%n==0)
        return n;
        return(GCD(n,m%n));
}
void main(){
    int m,n,o,p;
    printf("enter 4 num ");
    scanf("%d%d%d%d",&m,&n,&o,&p);
    printf("gcd of %d,%d,%d,%d is %d \n",
          m,n,o,p, GCD(GCD(m,n), GCD(o,p)));
}
```

Output:

```
Enter 4 num 45 54 90 999
gcd of 45, 54, 90, 999 is 9
```

Write a C program to print pattern

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

```
#include<stdio.h>
void main(){
  int n,i,j; int space;
  printf("Enter no.of rows: ");
  scanf("%d",&n);
  for(i=1; i<=n; i++){   if(i%2!=0){
    for(space=1; space<=n-i; space++){
      printf(" ");
    }
    for(j=1; j<=2*i-1; j++){
      if(j%2!=0)
        printf("*");    //* for odd places
      else
        printf(" ");    // space for even
    }
    printf("\n");
  }}
}
```

Output: Enter no.of rows: 5

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

Program to print Armstrong numbers in given range.

```
#include<stdio.h>
int isArmstrong(int x){
    int y=x;r,s=0;
    while(x!=0){
        r=x%10;
        s+=(r*r*r);
        x/=10;
    } if(y==s)
            return 1;
            return 0;
}
void main(){
 int n,start,end;
 printf("Enter range (start & end)(start<end): ");
 scanf("%d%d",&start,&end);
 printf("Armstrong no's b/w %d & %d are \n",start,end);
 for(int num = start; num<=end; n++){
     if(isArmstrong(num))
          printf("%d  ",num);
  }
}
```

Output: Enter range (start & end)(start <end): 1 1000

Armstrong no's b/w 1 and 1000 are:

1 153 370 371 407

Write program to print palindrome no's in given range.

```
#include<stdio.h>
int isPalindrome(int x){
    int y=x, rev=0, rem;
    while(x!=0){
        rem=x%10;
        rev=rev*10+rem;
        x/=10;
    }
    if(y==rev)
        return 1;
        return 0;
}
void main(){
    int n,start,end;
    printf("Enter range (start & end)(start<end): ");
    scanf("%d%d", &start,&end);
    printf("palindrome no's b/w %d & %d are:\n",
                                    start,end);
    for(int num=start; num<=end; num++){
        if(isPalindrome(num)){ printf("%d ", num);}}}
```

Output: Enter range (start &end)(start<end): 500 550

Palindrome no's b/w 500 & 550 are:

505 515 525 525 535 545

Write a program to print perfect numbers in given range.

```
#include<stdio.h>
int isperfect(int n){
    int i, s=0;
    for(i=1; i<n/2; i++)
        if(n%i==0)
            s+=i;
    if(s==n)
        return 1;
    return 0;
}
void main(){
    int num, start, end;
    printf("Enter the range(start & end)(start<end): ");
    scanf("%d %d", &start, &end);
    printf("perfect nos blw %d & %d are: \n", start, end);
    for(num=start; num<=end; num++)
        if(isperfect(num))
            printf("%d ", num);
}
```

Output: Enter the range (start & end)(start<end): 1 10000

6 28 496 8128

Write a program to find m^n.

```
#include<stdio.h>
int power(int m, int n){
    int i, p;
    for(p=1, i=1; i<=n; i++)
        p*=m;
    return p;
}
Void main(){
    int m,n;
    printf("Enter base & exponent value: ");
    scanf("%d %d", &m, &n);
    printf("%d^%d is: %d \n", m,n, power(m,n));
}
```

Output:

```
Enter base & exponent value: 2  10
2^10 is: 1024
```

Write a program to find sum of digits in a number.

```
#include<stdio.h>
void main(){
  int n, r, ch=1;
  while(ch==1){
     int s=0;
     printf("enter a number : ");
     scanf("%d", &n);
     while(n!=0){
        r=n%10;
        s+=r;
        n/=10;
  } printf("Sum of digits in %d is: %d ", n,s);
  printf("\n enter 1 to find sum of digits of one more
        number orelseenter 0 ");
  scanf("%d",&ch);
  }
}
```

Output: Enter a number: 78
Sum of digits in 78 is 15
Enter 1 for one more orelse enter 0 1
Enter a number: 678
Sum of digits in 678 is 21
Enter 1 for more orelse enter 0 0

Write a program to find reverse of a given number.

```
#include <stdio.h>
void main{
  int n, rem, rev, ch=1;
  while(ch==1){
    rev=0;
    printf("enter num to be reversed: ");
    scanf("%d", &n);
    while(n!=0){
      rem=n%10;
      rev=rev*10+rem;
      n/=10;
    }
    printf("Reverse is: %d", rev);
    printf("\n enter 1 for more or 0 to exit:");
    scanf("%d", &ch);
  }
}
```

Output: Enter num to be reversed: 678

Reverse is: 876

Enter 01 for more or 0 to exit: 1

Enter num to be reversed: 34567

Reverse is: 76543

Enter 1 for more or 0 to exit: 0

Write a program to perform an arithmetic operation given by user on numbers given by user using switch.

```
#include<stdio.h>
void main(){
   int num1, num2, result;
   char op;
   printf("Enter two numbers & an operator : ");
   scanf("%d %d %c", &num1, &num2, &op);
  int valid = 1;
  switch(op){
    case '+';
        result = num1 + num2;   break;
    case '-';
        result = num1-num2;   break;
    case '*':
        result= num1 * num2;   break;
    case '/':
        if(num2==0){
          printf("Div by 0 isn't possible \n");
          valid =0;
        }else
        result= num1 / num2;
        break;
```

```
        case '%':
            if(num2==0){
                printf("Modulus by 0 isn't possible\n");
                valid=0;
            }else{
                result = num1 % num2;
            }
            break;
        default:
            printf("Enter valid operator -> +, -, *, /, % \n");
            valid=0;
    }
    if(valid){
        printf("%d %c %d = %d \n", num1, op, num2, result);
    }
}
```

Output:

```
Enter two numbers & an operator : 5 3 0
Division by zero isn't possible
```

Output:

```
Enter two numbers & an operator: 5 5 *
5*5=25
```

Write a program to read 2 sets & find its intersection.

```
#include <stdio.h>
void main(){
   int A[10], B[10], m, n, i, C[10], j, K=0;
   printf("Enter set A size: ");
   scanf("%d", &m);
   printf("Enter set B size: ");
   scanf("%d", &n);
   printf("Enter set A elements: );
   for(i=0; i<m ; i++)
         scanf("%od ; & A[i]);
   printf("Enter set B elements:); for(i=0; i<n; i++)
         scanf("%d", & B[i]);
   for(i=0; i<m; i++){
   for(j=0; j<n; j++){
   if (A[i] == B[j])
      break;
   if (j<n)
      C[k++]= A[i];
   }
}
printf("A intersection B is: \n");
for(i=0, i<k; i++)
```

```
    printf("%d \t", C[i]);
}
```

Output:

```
Enter set A size: 7
Enter set B size: 6
Enter set A elements: 3  6  9  120 15  18  21
Enter set B elements: 5  10  15 120  25  30
A intersection B is:  15  120
```

Write a program to read two sets & find their Union.

```
#include<stdio.h>
int main(){
    int A[10], B[10], m,n,i, C[10],j, K=0;
    printf(" Enter size of A: ");
    scanf("%d", &m);
    printf(" Enter size of B: ");
    scanf("%d", &n);
    printf (" Enter set A elements: "); for(i=0; i<m; i++)
    scanf ("%d", &A[i]);
    printf (" \n Enter B set elements: ");
    for (i=0; i<n; i++)
    scanf("%d", &B[i]);
    for(i=0; i< m;i++){
    //for (j=0; j<n;j++) {
        // if (A[i] == [B[j]])
            // break;
    C[i] = A[i];
    }
    K=m;
    for( i=0; i<m;i++) {
    for (j=0; j<m ;j++)
        if[B[i] == A[j]
            break;
```

```
        if(j>=m)
            C[k+i] = B[i];
    }
    printf("A union B is:\n");
    for(i=0;i<k;i++)
    printf("%d\t", C[i]);
    return 0;
}
```

Output:

```
Enter size of A:
3
Enter size of B:
4
Enter set A elements:
12
24
36
Enter set B elements:
12
45
36
14
A union B is
12  24  36  45  14
```

Write a program to demonstrate call by value & Reference.

```
#include <stdio.h>
int example1(int x, int *y, int *z, int a){
    int m;
    printf("inside the function \n");
    printf("x=%d, y=%d, z=%d, a=%d \n",
                                x, *y, *z, a);
    x+=3;
    *y+=6;
    *z-=3;
    a*=2;
    m=x+(*y)+(*z)+a;
    printf("x=%d, y=%d, z=%d, a=%d, m=%d \n",
                                x, *y, *z, a, m);
    return m;
}
void main(){
int x=10, y=15, z=20, a=25, m;
printf("in main \n");
printf("x=%d, y=%d, z=%d, a=%d \n", x, y, z, a);
//pass the addresses of y and z
m=example1(x, &y, &z, a);
printf("back to main \n");
```

```
    printf("x=%d, y=%d, z=%d, a=%d, m=%d \n",
            x,y,z,a,m);

}
```

Output:

in main

x=10, y=15, z=20, a=25

inside the function

x=10, y=15, z=20, a=25

x=13, y=21, z=17, a=50, m=101

back to main

x=10, y=21, z=17, a=25, m=101

Write a program to perform matrix multiplication using array.

```
#include <stdio.h>
int main(){
    int i,j,k;
    int row1, column1, row2, column2;
    printf("Enter row1 & column1: ");
    scanf ("%d %d", &row1, &column1);
    int matrix1[row1][column1];
    printf("Enter Matrix1: \n");
    for(i=0; i<row1; i++){
        for(j=0; j<column1; j++){
            scanf("%d", & matrix1[i][j];
        }
    }
    printf (" Matrix 1: \n");
    for (i=0; i<row1; i++){
    for (j=0; j< column1; j++){
        printf ("%d", matrix1[i][j]);
    }
    printf ("\n");
    }
// printf ("Enter matrix 2: \n");
// for(i=0; i<row2; i++){
```

```
//for(j=0; j<column2; j++)
  printf("Enter row2 and column2: ");
  scanf("%d %d", &row2, &column2);
  int matrix2[row2][column2];
  printf("Enter Matrix2: \n");
  for(i=0 ; i< row2; i++){
  for(j=0; j<column2; j++){
  scanf("%d", &matrix2[i][j]);
  }
  }
  printf("Matrix2: \n")
  for(i=0 ;i<row2; i++){
  for(j=0; j<column2; j++){
     printf("%d" , matrix2[i][j]);
     }
     printf("\n");
  }
  if(column1 ==row2){
    int result[row1][column2];
    for(i=0; i<row1 ; i++){
      for(j=0; j< column2 ; j++){
         result[i][j] =0;
      }
    }
```

```
    for(i=0; i<row1; i++){
        for(j=0; j<column2; j++){
            for(k=0; k<column1; k++){
                result[i][j] += matrix1[i][k] * matrix[k][j];
            }
        }
    }
    printf("Result matrix: \n");
    for(i=0; i<row1; i++){
        for(j=0; j<column2; j++){
            printf("%od", result[i][j]);
        }
        printf("\n");
    }
    }
} else{
        printf("Matrix multiplication is not possible\n");
    }
    return 0;
}
```

Output ⇒

```
Enter row1 and clo column1: 2
 2
```

```
Enter Matrix 1:
1
2
3
4
Matrix 1:
1  2
3  4
Enter row 2 and column 2: 2
1
Enter Matrix 2:
1
2
Matrix 2
 1
 2
Resultant Matrix:
5
11
```

Write a program to add two matrices.

```
#include<stdio.h>
int main(){
    int r,c,a[100][100], b[100][100], sum[100][100], i,j ;
    printf("Enter row1, column1 of matrix1: ");
    scanf("%d %d", &r,&c);
    printf("\n Enter matrix1: \n");
    for(i=0;i<r; ++i){
        for(j=0; j<c;  ++j){
         //printf("Enter");
          scanf("%d",&a[i][j]);
      }
    }
    printf("\n Enter matrix2: \n");
    for (i=0; i<r; i++) {
        for(j=0; j<c; j++){
            scanf("%d", & b[i][j]);
        }
    }
    for(i=0; i<r ; i++)
        for(j=0; j<c; j++){
            sum[i][j] = a[i][j]+ b[i][j];
        }
    printf(" \n Sum of two matrices: \n");
```

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

Output:

```
Enter row, column of matrix: 2 2
Enter matrix 1:
1
2
3
4
Enter matrix 2:
5
6
7
8
Sum of two matrices:
   6   8
  10  12
```

Write a program to perform all operations on stack using array.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
struct Stack{
    int arr[MAX];
    int top;
};
void initStack(struct Stack *s){
    s->top = -1;
}
int isEmpty(struct Stack *s){
    return s->top == -1;
}
int isFull(struct Stack *s){
    return s->top == MAX-1;
}
void push(Struct Stack *s, int e){
   if(isFull(s)){
      printf("Stack overflow! can't push %d \n" e);
      return;
   }
   s->arr[++(s->top)] = e;
```

```
    printf("%d pushed to stack \n", e);
}
int pop(struct Stack *s){
    if(isEmpty(s)){
        printf("Stack Underflow! can't pop \n");
        return -1;
    }
    int e=s->arr[(s->top)--];
    printf("%d popped from stack \n", e);
    return e;
}
void display(struct Struct *s){
    if(isEmpty(s)){
        printf("Stack is empty \n");
        return;
    }
    printf("Stack elements: ");
    for(i= s->top ; i >=0; i--){
        printf("%d", s->arr[i]);
    }
    printf("\n");
}
int main()
{
```

```
struct Stack s;
initStack(&s);
int choice, value;
while(1){
    printf(" Stack Operations \n");
    printf(" 1. Push \n 2. Pop \n3. Display \n 4. Check if
        Empty \n 5. Exit \n Enter your choice : ");
    scanf ("%d", &choice);
    switch(choice){
    case 1:
        printf(" Enter value to push : ");
        scanf ("%d", &value);
        push (&s, value);
        break;
    case 2:
        pop(&s);
        break;
    case 3:
        display(&s);
        break;
    case 4;
      if (isEmpty(&s))
          printf(" Stack is Empty \n");
```

```
        else
            printf("Stack is not Empty \n");
        break;
    case 5:
        printf("Exiting program... ");
        return 0;
    default:
        printf("Invalid Choice! Try again \n");
    }
}
  return 0;
}
```

Output:

```
Stack operations:
1. Push
2. Pop
3. Display
4. Check if empty
5. Exit
Enter your choice: 1
Enter value to push: 10
10 pushed to stack
Enter your choice 3
Stack elements: 10
```

Enter your choice 2

10 popped from stack

Enter your choice 4

Stack is empty

Enter your choice 5

Exiting program...

Write a program to perform all operations on queue using arrays.

```
#include<stdio.h>
#include <stdlib.h>
#define MAX 5
struct Queue{
    int arr[MAX];
    int front, rear;
};
void initQueue(struct Queue *q){
    q->front = -1;
    q->rear = -1;
}
int isEmpty(struct Queue *q){
    return q->front == -1;
}
int isFull(struct Queue *q){
    return q->rear == MAX-1;
}
void enqueue(struct Queue *q, int e){
if(isFull(q)){
    printf("Queue overflow! can't enqueue %d \n", e);
    return;
 }
```

```
    if(isEmpty(q)){
        q->front=0;
    }

    q->arr[++(q->rear)]=e;
    printf("%d enqueued to queue\n",e);
}
int dequeue(struct Queue *q){
    if(isEmpty(q)){
        printf("Queue underflow! Can't dequeue\n");
        return -1;
    }
    int e=q->arr[q->front];
    if(q->front==q->rear){ //if there is only one element
        initQueue(q);
    }else{
        q->front++;
    }
    printf("%d dequeued from queue\n",e);
    return e;
}
void display(struct Queue *q){
    if(isEmpty(q)){
        printf("Queue is empty\n");
        return;
```

```
    }
    printf("Queue elements: ");
    for (int i = q->front; i<=q->rear; i++){
        printf("%d", q->arr[i]);
    }
    printf("\n");
}
int main(){
    struct Queue q;
    initQueue(&q);
    int choice, value;
    while(1){
        printf("\n Queue Operations:\n");
        printf(" 1.Enqueue \n 2. Dequeue \n 3. Display \n
              4. Check if Empty \n 5. Exit \n Enter
              your choice \n");
        scanf("%d", &choice);
        switch(choice){
        case 1:
            printf("Enter value to Enqueue: ");
            scanf("%d", &value);
            enqueue(&q, value);
            break;
```

```
        case 2:
            dequeue(&q);
            break;
        case 3:
            display(&q);
            break;
        case 4:
            if(isEmpty(&q))
                printf("Queue is Empty \n");
            else
                printf("Queue is not empty \n");
            break;
        case 5:
            printf("Exiting program...");
            return 0;
        default:
            printf("Invalid choice! Try again.\n");
    }
  }
  return 0;
}
```

Output:

```
Queue Operations:
```

1. Enqueue
2. Dequeue
3. Display
4. Check if empty
5. Exit

Enter your choice: 1
Enter value to enqueue: 10
10 enqueued to queue
Enter your choice: 1
Enter value to enqueue: 20
20 enqueued to queue
Enter your choice: 3
Queue elements: 10 20
Enter your choice: 2
10 dequeued from queue
Enter your choice: 5
Exiting program...

Write a program to perform all operations on stack using linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct node{
    int data;
    struct node *link;
};
int isEmpty(struct node *top){
    return top == NULL;
}
void push(struct node **top, int e){
    struct node *nnode = (struct node*) malloc(sizeof
                                    (struct node));
    nnode -> data = e;
    nnode -> link = *top;
    *top = nnode;
    printf("%d pushed to stack \n", e);
}
int pop(struct node **top){
  if(isEmpty(*top)){
    printf("Stack Underflow ! can't pop \n");
    return -1;
  }
```

```
    struct node *temp = *top;
    int e = temp -> data;
    *top = (*top -> link);
    free(temp);
    printf("%d popped from stack \n", e);
    return e;
}

void display(struct node *top){
 if(isEmpty(top)){
      printf("Stack is empty\n");
      return 0;
}

printf("Stack elements: ");
while(top){
    printf("%d -> ", top -> data);
    top = top -> link;
    }
    printf("NULL\n");
}

int main(){
struct node *top = NULL;
int choice, value;
while(1){
```

```
printf("\n Stack operations \n");
printf(" 1.Push \n 2.Pop \n 3.Display \n 4.Check if Empty \n
        5.Exit \n");
printf(" Enter ur choice : ");
scanf("%d", &choice);
switch(choice){
   case 1:
        printf(" Enter value to push: ");
        scanf("%d~~value~~", &value);
        push(&top, value);
        break;
  case 2:
        pop (&top);
        break;
  case 3;
        display(top);
        break;
  case 4:
     if(isEmpty(top))
          printf ("Stack is Empty \n");
     else
          printf("Stack is not empty \n");
     break;
```

```
        case 5:
            printf("Exiting program...\n");
            return 0;
        default:
            printf("Invalid choice! Try again.\n");
    }
}

    return 0;
}
```

Output:

```
Stack Operations:
1. Push
2. Pop
3. Display
4. Chek if Empty
5. Exit
Enter your choice: 1
Enter value to push: 10
10 pushed to stack
Enter your choice: 01
Enter value to push: 20
20 pushed to stack
Enter your choice: 3
```

```
stack elements: 20 -> 10 -> NULL
Enter your choice: 2
20 popped from stack
Enter your choice: 4
Stack is not empty
Enter your choice: 5
Exiting programm...
```

Write a program to perform all operations on queue using linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct node{
        int data;
        struct node *link;
};
struct Queue{
    struct node *front;
    struct node *rear;
};
void initQueue(struct Queue *q){
     q->front = q-> rear =NULL;
}
int isEmpty(struct Queue *q){
     return q->front ==NULL;
}
void enqueue(struct Queue *q, int e){
    struct node *nnode =(struct *nnode) malloc(sizeof
                                        (struct node));
    nnode ->data =e;
    nnode -> link = NULL;
```

```
    if(q->rear==NULL){ // if queue is empty
        q->front = q->rear = nnode;
    }else{
        q->rear->link = nnode;
        q->rear = nnode;
    }
    printf("%d enqued to queue\n", e);
}
int dequeue(struct Queue *q){
    if(isEmpty(q)){
        printf("Queue Underflow! can't dequeue\n");
        return -1;
    }
    struct node *temp = q->front
    int e = temp->data;
    q->front = q->front->link;
    if(q->front==NULL){ // if Q becomes empty after deque
        q->rear = NULL
    }
    free(temp);
    printf("%d dequeued from queue\n",e);
    return e;
}
```

```
void display(struct Queue *q){
    if(isEmpty(q)){
        printf("Queue is empty\n");
        return;
    }
    struct node *temp = q->front;
    printf("Queue elements: ");
    while(temp){
        printf("%d -> ", temp->data);
        temp = temp->link;
    }
    printf("NULL\n");
}
int main(){
    struct Queue q;
    initQueue(&q);
    int choice, value;
    while(1){
        printf("\n Queue Operations:\n");
        printf("1. Enqueue\n 2. Dequeue\n 3. Display\n
                4. Check if Empty\n 5. Exit\n");
        printf("Enter your choice\n");
        scanf("%d", &choice);
```

```
switch(choice){
case 1:
    printf("Enter value to enqueue: ");
    scanf("%d",&value);
    enqueue(&q, value);
    break;
case 2:
    dequeue(&q);
    break;
case 3:
    display(&q);
    break;
case 4:
  if(isEmpty(&q))
      printf("Queue is Empty \n");
  else
      printf("Queue is not empty \n");
  break;
case 5:
  printf("Exiting program...\n");
  return 0;
default:
    printf("Invalid choice! Try again :) \n");
}
```

```
    }
    return 0;
}
```

Output:

```
Queue Operations:
1. Enqueue
2. Dequeue
3. Display
4. Check if Empty
5. Exit
Enter your choice: 1
Enter value to enqueue: 10
10 enqueued to queue
Enter your choice: 3
Queue elements: 10 -> NULL
Enter your choice: 2
10 dequeued from queue
Choice : 4
Queue is not Empty
Enter your choice: 5
Exiting program ...
```

Write a ~~perf~~ program to perform operations on circular queue on linked list.

```
#include<stdio.h>
#include<stdlib.h>
#define M4
struct Node{
    int data;
    struct Node *next;
}
struct CircularQueue{~~Front~~
    struct Node *front;
    struct Node *rear;
}
void initQueue(struct CircularQueue *q){
    q->front = q->rear = NULL;
}
int isEmpty(struct CircularQueue *q){
    return q->front ==NULL;
}
void enqueue(struct CircularQueue *q, int e){
    struct Node *nnode=(struct Node*) malloc(sizeof
                                    (struct Node));
    nnode->data =e;
    nnode->next = NULL;
```

```
    if(isEmpty(q)){
        q->front = q->rear = nnode;
        q->rear->next = q->front;   //make it circular
    }else{
        q->rear->next = nnode;
        q->rear = nnode;
        q->rear->next = q->front;   //maintain circular link
    }
    printf("%d enqueued to queue\n", e);
}
int dequeue(struct CircularQueue *q){
    if(isEmpty(q)){
        printf("Queue Underflow! Can't dequeue\n");
        return -1;
    }
    int e;
    struct Node *temp = q->front;
    e = temp->data;
    if(q->front == q->rear){        //if only 1element
        q->front = q->rear = NULL;
    }else{
        q->front = q->front->next;
        q->rear->next = q->front;
    }
```

```
    free(temp);
    printf("%d dequeued from queue\n", e);
    return e;
}

void display(struct CircularQueue *q){
    if(isEmpty(q)){
        printf("Queue is empty\n");
        return;
    }
    struct Node *temp = q->front;
    printf("Queue elements: ");
    do{
        printf("%d", temp->data);
        temp = temp->next;
    }while(temp != q->front);
    printf("\n");
}

void main(){
    struct CircularQueue q;
    initQueue(&q);
    int choice, value;
    while(1){
        printf("\n CircularQueue Operations:\n 1. Enqueue\n
```

```
2. Dequeue \n3. Display \n 4. Check if Empty \n
5. Exit \n Enter your choice: ");
scanf("%d", &choice);
switch(choice){
case 1:
    printf("Enter value to enqueue: ");
    scanf("%d", &value);
    enqueue(&q, value);
    break;
case 2:
    dequeue(&q);
    break;
case 3:
    display(&q);
    break;
case 4:
    if (isEmpty(&q))
        printf("Queue is empty\n");
    else
        printf("Queue is not empty\n");
    break;
case 5:
    printf("Exiting program...\n");
    return 0;
```

```
        default:
            printf("Invalid choice! Try again\n");
        }
    }
}
```

Output:

```
Circular Queue Operations
1. Enqueue
2. Dequeue
3. Display
4. Check if empty
5. Exit
Enter your choice: 1
Enter value to enqueue: 10
10 enqueued to queue
Enter your choice: 3
Queue elements: 10
Enter your choice: 2
10 dequeued from queue
Enter your choice: 4
Queue is empty
Enter your choice: 5
Exiting program...
```