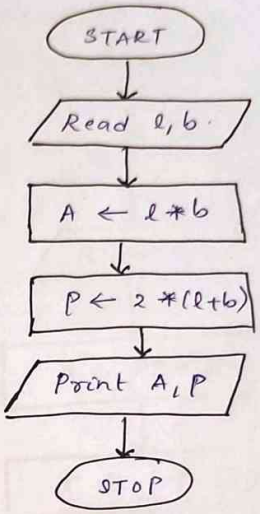
WAP to find area and perimeter of rectangle. along with an algorithm and flowchart.

algorithm:

Steps:

① START
② Read l, b
③ A ← l * b
④ P ← 2 * (l + b)
⑤ Print A, P
⑥ STOP.



program:

```
#include <stdio.h>
void main() {
    int l, b, a, P;
    printf ("Enter values for length & breadth:");
    scanf ("%d %d", &l, &b);
    a = l * b;
    P = 2 * (l + b);
    printf ("The area of rectangle : %d \n", area);
    printf ("The perimeter of rectange : %d \n", p);
}
```

OUTPUT:

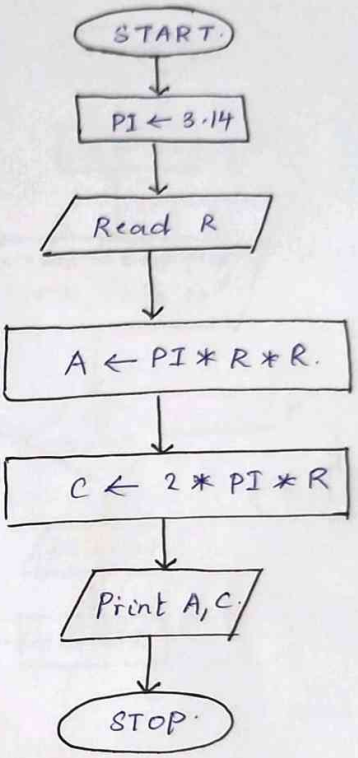
Enter values for length and breadth : 5 3
The area of rectangle : 15
The perimeter of rectangle : 16

Write an algorithm, flowchart and program to calculate area and circumference of a circle.

Algorithm:

1. START
2. pi ← 3.14.
3. Read radius.
4. A ← pi * radius * radius
5. C ← 2 * pi * radius.
6. print A, C.
7. STOP.

Flowchart:



program:

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

OUTPUT:

Enter circle's radius : 5.3

area : 88.202606

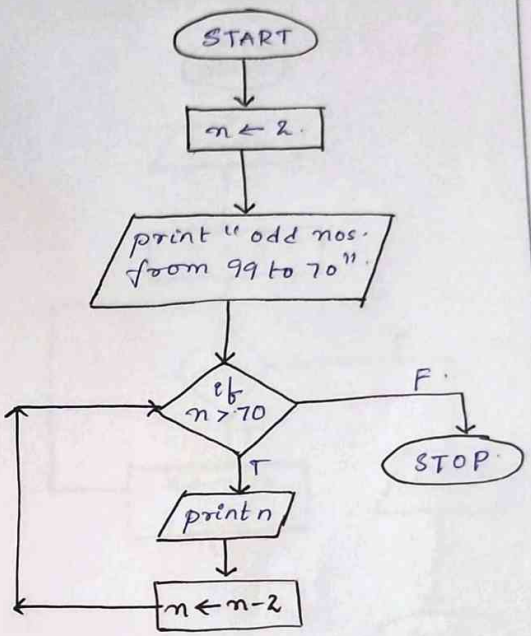
circumference : 33.284000

Write an algorithm, flowchart and a program to print odd numbers from 99 to 70.

Algorithm:

1. START
2. $n \leftarrow 99$
3. Print "odd nos. from 99 to 70: ".
4. if $n >= 70$ goto (5) else goto (8)
5. print n
6. $n \leftarrow n-2$
7. goto (4)
8. STOP

Flowchart:



program:

```
#include <stdio.h>
void main() {
        int n = 99;
        printf ("odd nums. from 99 to 70: \n");
    while (n > 70) {
            printf ("%d", n);
            n -= 2;
    }
}
```

OUTPUT:

Odd nums. from 99 to 70:

99 97 95 93 91 89 87 85 83 81 79

77 75 73 71

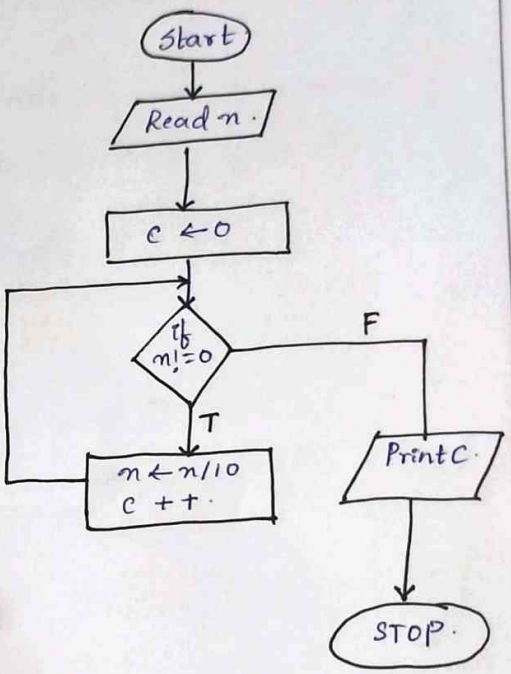
Write an algorithm, draw a flowchart and write a program to read a number n and find the number of digits it has.

Algorithm :

STEPS:

1. START
2. Read n
3. c ← 0
4. if (n!=0) goto step 5 else goto step 8
5. n ← n/10
6. c ← c + 1.
7. goto step 4
8. print c.
9. STOP.

Flowchart::



Program :

```
#include <stdio.h>
void main()
{
    int n, c=0;
    printf ("Enter a number: \n");
    scanf ("%d", &n);
    while (n!=0)
    {
        n= n/10;
        c++;
    }
    printf ("number of digits are : %d", c);
}.
```

OUTPUT:

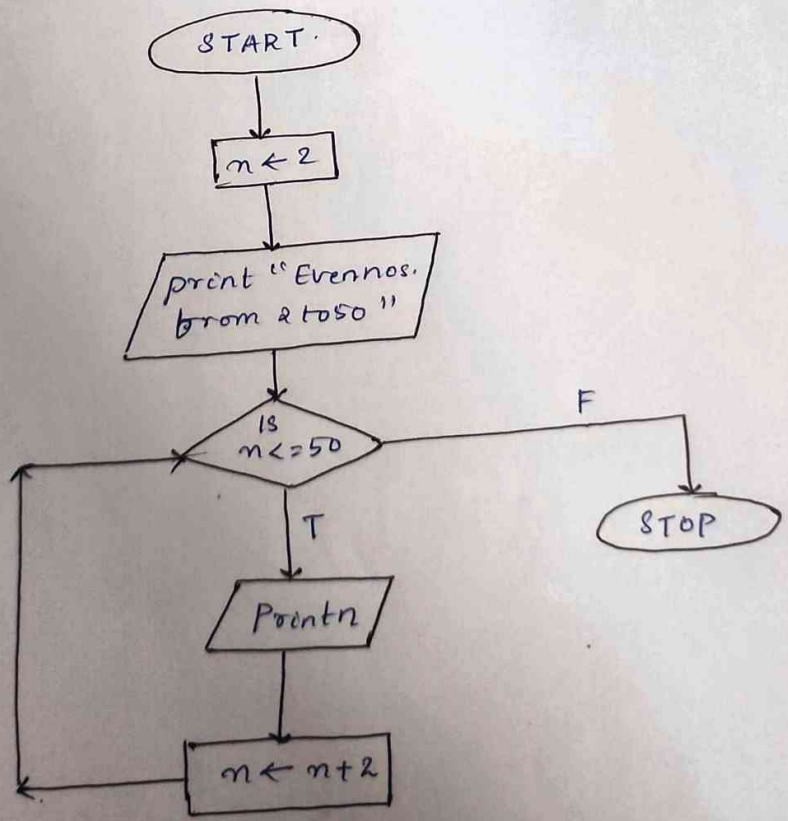
Enter a number : 365
Number of digits are : 3.

WAP, an algorithm & flow chart to print even numbers from 2 to 50.

Algorithm:-

1. START
2. Read $n \leftarrow 2$
3. print "Even nos. from 2 to 50: \n";
4. If $n <= 50$ goto (5) else goto (8)
5. print n
6. $n \leftarrow n + 2$
7. goto (4)
8. STOP.

Flowchart:-



program:

```
#include <stdio.h>
void main() {
    int n = 2;
    printf ("Even nos. b/w 2 to 50 : \n");
    while (n<=50) {
        printf ("%d", n);
        n+=2;
    }
}
```

OUTPUT:

Even nos. b/w 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

WAP to find the biggest of any three given numbers; also write an algorithm & flowchart.

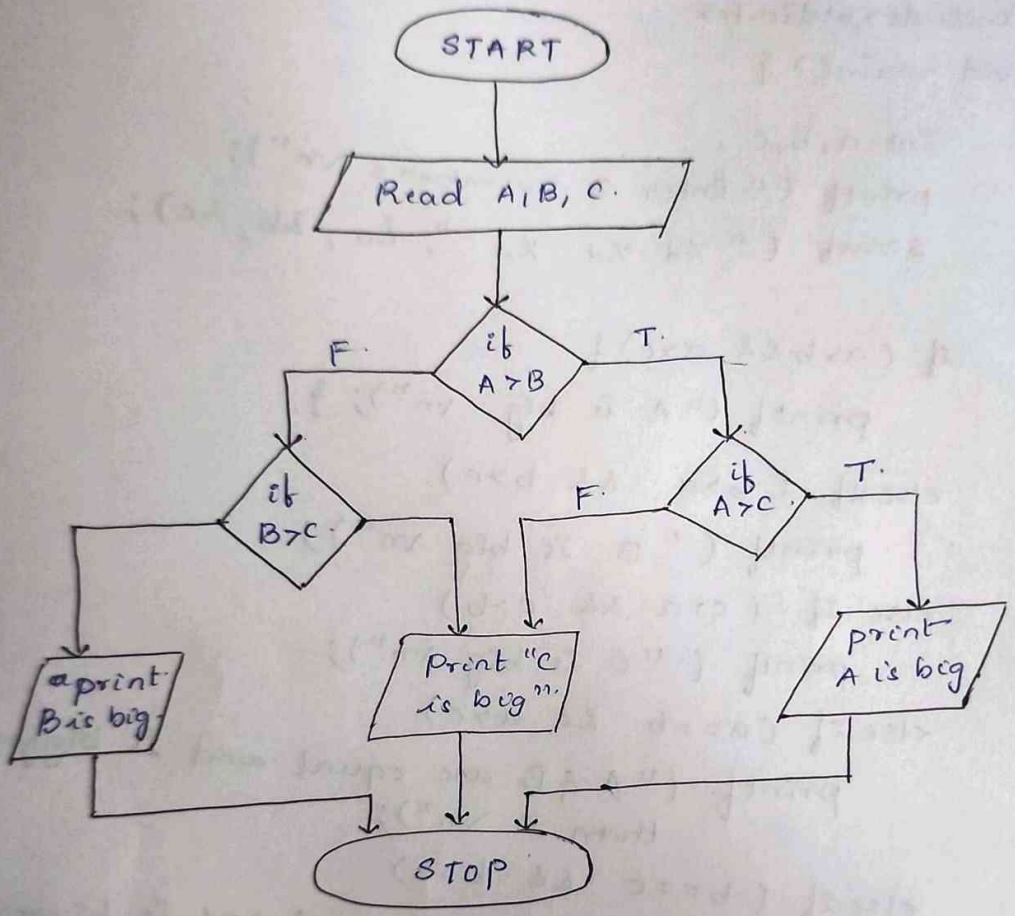
program:

```
#include<stdio.h>
void main() {
    int a,b,c ;
    printf ("Enter 3 numbers \n");
    scanf ("%d %d %d ", &a, &b, &c);

  if (a>b && a>c) {
     printf ("A is big \n"); }.
  else if (b>a  && b>c)
     printf ("B is big \n");
  else if (c>a && c>b)
     printf ("C is big \n");
  else if (a==b  && a>c)
     printf ("A & B are equal and is bigger
              than c \n");
  else if (b==c  && b>a)
     printf ("B & C are equal and is bigger
              than A \n");
  else if (a==c  && a>b)
     printf ("A & C are equal and is bigger
              than B \n");

  else
     printf ("A, B, C are equal \n");
```

OUTPUT :

```
Enter 3 numbers :
 45   60   105
C is big.
```

FLOWCHART:


START
Read A, B, C.
if A>B
F.
T.
if B>C.
if A>C.
F.
T.
print B is big
Print "C is big".
print A is big
STOP

Algorithm:2.

1. START.
2. Read a, b, c.
3. if a>b && a>c goto (4) else goto (5)
4. print "A is big".
5. STOP.
6. if b>a && b>c goto (7) else goto (9)
7. print "B is big".
8. STOP
9. if c>a && c>b goto (10) else goto (12)
10. print c is big.
11. STOP
12. if a==b && a>c goto (13). else goto (15)
13. print a, b are equal & bigger than c
14. STOP
15. if b==c && b>a goto (16) else goto (18)
16. print B, C are equal & bigger than A.
17. STOP
18. if c==a && a>b goto (19) else goto (21)
19. print C, A are equal and bigger than B.
20. STOP
21. Print "All are equal"
22. STOP.

WAP to display the first ten natural numbers using for loop.

```
#include <stdio.h>
int main()
{
    int i, n = 10;
    printf("the first 10 natural numbers: \n");
    for(i=1; i<=n; i++)
    {
        printf("%d \n", i);
    }
}
```

OUTPUT:

the first 10 natural numbers are :

1
2
3
4
5
6
7
8
9
10

Write a program to find the sum of digits of a given number.

```
#include <stdio.h>
    int main() {
        int n, sum = 0, digit;
    printf ("Enter a number: ");
    scanf ("%d", &n);

    while (n != 0) {
            digit = n % 10;
            sum += digit;
              n /= 10;
        }
    printf ("Sum of digits = %d \n", sum);
        return 0;
    }
```

OUTPUT:

```
Enter a number : 1234
Sum of digits = 10
```

WAP to display the sum of natural numbers using while loop.

```
#include <stdio.h>
int main()
{
    int n, i, count = 1, sum = 0;
    printf("Enter number of natural numbers\n");
    scanf("%d", &n);
    printf("The 1st ten natural numbers are:\n");
    for(i=1; i<=n; i++)
    {
        printf("%d \t", i);
    }
    while(count<=n)
    {
        sum = sum + count;
        count++;
    }
    printf("Sum of first %d natural numbers is %d", n, sum);
    return 0;
}
```

OUTPUT:

```
Enter number of natural numbers:
10
The first ten natural numbers are:
1  2  3  4  5  6  7  8  9  10
Sum of first 10 natural numbers is 55
```

WAP to print a table of number 2 using do-while loop.

```
#include <stdio.h>
int main()
{
    int num = 1 , table;
    printf ("Displaying the multiples of 2 \n");

    do
    {
        table = 2 * num;
        printf ("%d \n", table);
        num++;
    } while (num <= 10);

    return 0;
}
```

OUTPUT:

Displaying the multiples of 2 :

2
4
6
8
10
12
14
16
18
20

WAP to find the factorial of number.

```
#include <stdio.h>
int main ()
{
 int i , fact = 1 , num ;
    printf (" Enter a number to find factorial ");
 scanf ( "%d ", & num);

 for (i= 1 ; i<=num ; i++)
 {
    fact = fact * i ;
 }
 printf ( " Factorial of %d is : %d ", num, fact);
 return 0;
}.
```

OUTPUT :

```
Enter a number to find factorial : 12
Factorial of 12 is : 479001600.
```

Write a program to display the n terms of harmonic series and their sum.

// series is 1 + 1/2 + 1/3 + 1/4 + ... + 1/n terms.

```
#include <stdio.h>
void main
{
  int i, n
  float s=0.0;
      printf ("Input the no. of terms : ");
      scanf ("%d", &n);
      printf ("\n\n");
   for(i=1, i<=n; i++)
   {
     if (i<n)
     {
        printf ("1 %d + ", i);
          s+ = 1/(float)i ;
      }
     if (i==n)
     {
       printf ("1 %d ", i);
       s+ = 1/(float)i;
      }
    }
printf("\n Sum of Series upto %d terms : %f \n", n,s);
 }
```

OUTPUT:

Input the number of terms : 5

1/1 + 1/2 + 1/3 + 1/4 + 1/5

Sum of series upto 5 terms : 2.283334

WAP to read n number of values in an array and display it in reverse order.

```
#include <stdio.h>
void main()
{
  int i, n, a[100];
      printf ("Read n number of values in an
        array and display it in reverse order\n");
      printf ("Enter the number of elements to
              store in the array : ");
      scanf ("%d", &n);
      printf ("Enter %d number of elements in
              array : \n", n);

    for (i=0; i<n ; i++)
    {
       printf ("element - %d : ", i);
       scanf ("%d ", &a[i]);
    }
printf ("\n the values store into the array
 are : \n");

        for (i=0; i<n; i++)
        {
            printf ("%5d ", a[i]);
        }
printf ("The values stored into array in
        reverse are : \n");

for (i=n-1 ; i>=0; i--)
   {
```

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

OUTPUT:

Read n number of values in an array and display it in reverse order.

```
Enter the number of elements to store in array: 3
Enter the number of elements in array:
 element-0 : 2
 element-1 : 5
 element-2 : 7.
The values stored in array are:
   2   5   7.
The values stored into array in reverse are:
   7   5   2
```

WAP to find maximum & minimum element in an array.

```
#include <stdio.h>
  void main()
  {
      int arr1[100]
      int i, mx, mn, n;
    printf (" Enter the number of elements to be
      stored in array : ");
    scanf ("%d", &n);
    printf (" \n \n");

          for (i=0; i<n ; i++)
          {
              printf (" element - %d ", i);
              scanf ("%d", &arr1[i]);

          }.

    mx = arr1[0]
    mn = arr1[0];

for (i= 1 ; i<n ; i++)
 {
     if (arr1[i] > mx)
      {
            mx = arr1[i];
       }

  if (arr1[i] < mn)
        mn = arr1[i];

 }
```

```
printf ("Maximum element is : %d \n", mx);
printf ("Minimum element is : %d \n", mn);
}.
```

OUTPUT :

```
Enter the number of elements to be stored
in array : 3
Enter 3 elements in array :

element - 0 : 45
element - 1 : 25
element - 2 : 30

Maximum element is : 45
Minimum element is : 25
```

WAP to print a palindrome number.

// a number n is a palindrome if the reverse of n is also n. eg: 121 = 121.

```
#include <stdio.h>
void main()
{
    int n, r, rev=0, m;
    printf("Enter a number to check : \n");
    scanf("%d", &n);
        m=n;
        while(n!=0)
        {
            r = n%10;
            rev = (rev*10)+r;
            n = n/10;
        }
    if(m==rev)
        printf("given %d is palindrome", m);
    else
        printf("given % is not a palindrome", m);
}
```

OUTPUT:

Enter a number to check :

1234321

Given 1234321 is palindrome.

Write a program to read a number and check if it is a perfect number or not.

// perfect no. = if sum of factors is equal to 2n.

```
#include <stdio.h>
void main()
  {
     int i, n, s=0;
   printf ("Enter a number : \n");
   scanf ("%d", &n);

   for(i=1; i<=n/2; i++)

        if (n%i==0)
           s=s+i;

(s==n)? printf ("given number is a perfect no."):
        printf ("given num isn't a perfect num");
return 0;
}
```

OUTPUT:

1. Enter a number :
   28
   Given number is a perfect number.

2. Enter a number :
   190
   Given number is not a perfect number.

WAP to print an armstrong number :-

// 'n' is armstrong if sum of cubes of digits of 'n' is the number itself. eg: $153 = 1^3 + 5^3 + 3^3$.

```
#include <stdio.h>
void main() {
  int n, r, m, s=0;
  printf ("Enter a number to check : ");
  scanf ("%d", &n);
  m = n ;
    while (n!=0)
      {
        r = n %. 10;
        s+ = (r*r*r);
        n = n/10;
      }
(s==n)? printf ("Armstrong \n") : printf("
    given num is not an armstrong num \n");
}.
```

OUTPUT :-

1. Enter a number to check :
   371
   The entered number is Armstrong number.

2. Enter a number to check :
   124321
   The entered number is not an Armstrong number.

WAP to implement sum of elements of an array using one dimensional array.

```
#include <stdio.h>
int main()
{
    int arr[5], i, s=0;
    for(i=0; i<5; i++)
    {
        printf("Enter a[%d]: ", i);
        scanf("%d", &arr[i]);
    }
    for (i=0; i<5; i++)
    {
      s += arr[i];
    }
  printf("\n sum of array element = %d", s);
  return 0;
}
```

OUTPUT:

```
Enter a[0]: 2
      a[1]: 5
      a[2]: 1
      a[3]: 7
      a[4]: 4
Sum of array elements = 19.
```

WAP to implement Addition of two matrices using 2-D array.

```
#include <stdio.h>
int main()
{
  int r, c, a[10][10], b[10][10], sum[10][10], i, j;
  printf("Enter num. of rows (between 1 & 10): ");
  scanf("%d", &r);
  printf("Enter num. of columns (between 1 & 10): ");
  scanf("%d", &c);
  printf("\n Enter elements of 1st matrix: \n");
    for(i=0; i<r; i++)
      for(j=0; j<c; j++).
       {
          printf("Enter element a %d %d", i+1,
                                          j+1);
          scanf("%d", &a[i][j]);
       }
printf("Enter elements of 2nd matrix: \n");
      for(i=0; i<r; i++)
        for(j=0; j<c; j++)
        {
          printf("Enter element b %d %d", i+1,
                                          j+1);
          scanf("%d", &b[i][j]);
        }

// adding 2 matrices
   for(i=0; i<r; i++)
     for(j=0; j<c; j++) {
         sum[i][j] = a[i][j] + b[i][j]; }.
```

```
// printing result:
    printf ("\n Sum of 2 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 number of rows (b/w 1 & 10) : 2
Enter number of columns (b/w 1 & 10) : 2.

Enter elements of 1st matrix :
enter element : a11 : 3
                a12 : 5
                a21 : 7.
                a22 : 4.
Enter elements of 2nd matrix :
enter element : b11 : 1
                b12 : 8
                b21 : 6
                b22 : 9

Sum of 2 matrices :   4  13.
                     13  13
```

(a) Program to implement Linear Search.

```
#include <stdio.h>
int main()
{
    int a[20], i, x, n;
    printf ("How many elements for array ?\n");
    scanf ("%d", &n);
    printf ("Enter array elements :\n");
    for ( i=0 ; i<n ; i++)
     {
         scanf ("%d", &a[i]);
     }
    printf ("Enter element to search : ");
    scanf ("%d", &x);
    for (i=0; i<n ; i++)
     {
         if (a[i] == x)
          break;
     }
    if (i<n)
     {
         printf ("Element found at index %d", i);
     }
    else
     { printf (" Element not found "); }.
     }
}
```

OUTPUT :

How many elements for array ?
5
Enter array elements : 2 5 8 4 9
Element to search : 4
Element found at index 3.

(a) Program to implement Linear Search.

```
#include <stdio.h>
int main()
{
    int a[20], i, x, n;
    printf ("How many elements for array ?\n");
    scanf ("%d", &n);
    printf ("Enter array elements : \n");
    for ( i=0; i<n ; i++).
     {
         scanf ("%d", &a[i]);
     }
    printf ("Enter element to search : ");
    scanf ("%d", &x);
    for (i=0; i<n ; i++).
     {
         if (a[i] == x)
         break;
     }
     if (i<n)
     {
        printf ("Element found at index %d", i);
     }
    else
     { printf (" Element not found "); }.
    }
}
```

OUTPUT:

```
How many elements for array ?
5
Enter array elements : 2 5 8 4 9
Element to search : 4
Element found at index 3.
```

(6) program to implement Binary Search.

```
#include <stdio.h>
int main()
{
    int c, first, last, middle, n, search, array[100];
    printf("Enter number of elements \n");
    scanf("%d", &n);
    printf("Enter %d integers : \n", n);
    for (c=0; c<n; c++)
    {
        scanf("%d", &array[c]);
    }
    printf("Enter value to find : \n");
    scanf("%d", &search);
    first = 0;   last = n-1;   middle = (first+last)/2;
    while (first <= last)
    {
        if (array[middle] < search)
        {
            first = middle + 1; }
        else if (array[middle] == search)
        {
            printf("%d found at location %d \n", search,
                                                 middle);
            break;
        }
        else{
            last = middle-1;
            }
        middle = (first + last)/2;
    }
    if (first > last) {
```

```
printf ("Not found! %d isn't present in list", search);
}
return 0;
}
```

OUTPUT:

```
Enter number of elements:
5
Enter 5 integers:
11
16
25
39
60
Enter value to find:
60
60 found at location 4.
```

WAP to implement structures to store information of students.

```
#include <stdio.h>
struct student
{
    char FirstName[50];
    int roll;
    float marks;
} s[10];

int main()
{
    int i;
    printf("Enter information of students:\n");
//storing information
    for(i=0; i<5; i++)
    {
        s[i].roll = i+1;
        printf("For roll number %d\n", s[i].roll);
        printf("Enter first name: ");
        scanf("%s", s[i].firstName);
        printf("Enter marks: ");
        scanf("%f", &s[i].marks);
    }
printf("Displaying information:\n\n");
for(i=0; i<5; i++)
  {
    printf("Roll number: %d", i+1);
    printf("First name: ");
```

```
        puts (s[i].firstName);
        printf ("Marks : %lf ", s[i].marks);
        printf ("\n");
        }
    return 0;
}
```

OUTPUT:

```
Enter information of students :

For roll number 1,
Enter first name: Sneha

Enter marks : 100

For roll number 2,
Enter first name : Gokul
Enter marks : 98

For roll number 3,
Enter first name : Sandhya
Enter marks : 90

Displaying information :

Roll number : 1
First name : Sneha
Marks : 100.0

Roll number : 2
First name : Gokul
Marks : 98.0

Roll number : 3
First name : Sandhya
Marks : 90.0
```

Program to implement structures and pointers.

```
#include<stdio.h>
struct person
{
   int age;
   float weight;
   char name[30];
};
int main()
{
    struct person *personPtr, person1;
    personPtr = &person1;
    printf("Enter name: \n");
    scanf("%s", &personPtr->name);
    printf("Enter age : \n");
    scanf("%d", &personPtr->age);
    printf("Enter weight : \n");
    scanf("%f", &personPtr->weight);

   printf("Displaying :\n");
   printf("Name : %s\n", personPtr->name);
   printf("Age : %d\n", personPtr->age);
   printf("weight : %f", personPtr->weight);

   return 0;
}
```

OUTPUT: Enter name

Sandhya

Enter age :

30

Enter weight :

65.55

Displaying :

Name : Sandhya Age : 30 Weight : 65.550003.

WAP to implement dynamic memory allocation.

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

struct person
{
    int age;
    char name[30];
};

int main()
{
    struct person *ptr;
    int i, n;
    printf("Enter number of persons: \n");
    scanf("%d", &n);
// allocating memory for n numbers of struct person
    ptr = (struct person*) malloc(n * sizeof(struct person));
for(i=0; i<n; i++)
{   printf("Enter name & age respectively: \n");
    scanf("%s %d", (ptr+i)->name, &(ptr+i)->age);
}
printf("Displaying information: \n");
    for(i=0; i<n; i++)
        printf("Name: %s \t Age: %d \n",
               (ptr+i)->name, (ptr+i)->age);

    return 0;
}
```

OUTPUT :

```
Enter number of persons:
3
Enter name and age respectively
Sandhya
25
Enter name and age respectively
karthik
30
Enter name and age respectively
Jayshree
35
```

Displaying information :

```
Name : Sandhya      Age : 25
Name : Karthik      Age : 30
Name : Jayshree     Age : 35
```

(a) WAP to implement stack operations.

```
#include <stdio.h>
#define MAXSIZE 5
struct stack
{
   int stk[MAXSIZE];
   int top;
};
typedefstruct stack STACK;
STACK s;
void push(void);
int pop(void);
void display(void);
void main()
{
   int choice;
   int option = 1;
   s.top = -1;
   printf("STACK operations \n");
   while(option)
   {
   printf("_____\n");
   printf("1. PUSH \n 2.POP\n, 3. DISPLAY \n
        4. EXIT \n");
   printf("Enter your choice:");
```

```
switch (choice)
{
    case 1 :
        push();
        break ;
    case 2 :
        pop();
        break;
    case 3 :
        display();
        break;
    case 4 :
        return ;
    }

fflush (stdin)
printf ("Do you want to continue (Type 0 or 1)?");
scanf ("%d", &option);
    }
}

/* Function to add an element to stack */

void push()
  {
    int num ;
    if (s.top == (MAXSIZE - 1))
    {
      printf ("Stack is Full.\n");
      return;
    }
  else
    {
      printf ("Enter elt to be pushed: ");
```

```
    scanf ("%d", &num);
    s.top = s.top+1;
    s.stk [s.top] = num;
   }
   return;
}

/* Function to delete an element from stack */

int pop()
{
   int num;
   if (s.top == -1)
   {
      printf ("Stack is Empty");
      return (s.top);
   }
   else
   {
      num = s.stk [s.top];
      printf ("popped element is = %d \n",
                                  s.stk [s.top]);
      s.top = s.top -1;
   }
   return (num);
}

/* Function to display the status of stack */

void display().
{
   int i;
```

```
if (s.top == -1)
 {
    printf ("stack is empty \n");
    return;
  }
else
   { printf (" The status of stack is\n");
 for (i=s.top ; i>=0 ; i--)
  {
    printf ("%d \n", s.stk [i]);
  }
}
```

STACK operations :

1. push 2. pop 3. display 4. exit.

Enter your choice : 1
Enter the element to be pushed : 3
Do you want to continue (0 or 1) ? 1

1. push 2. pop 3. display 4. exit

Enter your choice: 1
Enter element to be pushed: 4
Do you want to continue (type 0 or 1) ? 1

1. push 2 pop 3. display 4: exit

Enter your choice : 1.
Enter element to be pushed : 6
Do you want to continue (Type 0 or 1) ? 1.

1. push 2. pop 3. Display 4: exit.

Enter your choice : 3

The status of stack is

6

4

3

Do you want to continue (Type 0 or 1) ? 1

1. push 2. pop 3. display 4. exit

Enter your choice : 2

popped element is = 6

Do you want to continue (Type 0 or 1) ? 1

1. push 2. pop 3. display 4. exit

Enter your choice : 3

The status of stack is

4

3

Do you want to continue (type 0 or 1) ? 1

1. push 2. pop 3. Display 4. exit.

Enter your choice : 4.

Program to demonstrate Queue operations.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 50

void insert();
void delete();
void display();

int queue_array[MAX];
int rear = -1;
int front = -1;

int main()
{
    int choice;
    while(1)
    {
        printf("1. Insert element \n, 2. Delete element from queue \n", 3. Display all elements \n
4. Quit \n");
        printf("Enter your choice :");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1:
                insert();
                break;
            case 2:
                delete();
                break;
            case 3:
                display();
                break;
```

```
        case 4:
            exit(1);
            default:
                printf("wrong choice \n");
        }
    }
}
void insert()
{
    int add_item;
    if (rear == MAX-1)
    printf("Queue Overflow \n");
    else
    {
        if (front == -1)
            // if queue is initially empty
        front = 0;
        printf("Insert elt in queue: ");
        scanf("%d", &add_item);
        rear = rear + 1;
            queue_array[rear] = add_item;
    }
}
void delete()
{
    if (front == -1 || front > rear)
    {
        printf("Queue Underflow \n");
        return;
    }
```

```
else
 {
  printf ("Element deleted from queue is %d \n",
          queue_array [front]);
   front = front +1;
  }
 }
 void display ()
 {
    int i;
     if (front == -1)
       printf ("Queue is empty \n");
      else
       {
         printf ("Queue elts are: \n");
        for (i=front; i<=rear; i++)
           printf ("%d", queue_array[i]);
         printf ("\n");
       }
      }
```

OUTPUT:

1. Insert element to queue.
2. Delete element from queue.
3. Display all elements of queue
4. Quit.

Entere your choice: 3
Queue is Empty.

1. Insert 2. Delete 3. Display 4. Quit.

Enter your choice : 1.

Insert the element in queue : 6

1. Insert 2. Delete 3. Display 4. Quit.

Enter your choice : 1.

Insert the element in queue : 9

1. Insert 2. Delete 3. Display 4. Quit.

Enter your choice : 3.

Queue elements are

6 8 9

1. Insert 2. Delete 3. Display 4. Quit

Enter your choice : 2.

Queue elements are :

89

1. Insert 2. Delete 3. Display 4. Quit.

Enter your choice : 4.

WAP to demonstrate a stack using linked list.

```
#include <stdio.h>
#include <stdlib.h>
 struct node.
 {
     int info;
    struct node *ptr;
 } *top, *top1, *temp;
int topelement();
   void push(int data);
   void pop();
   void empty();
    void display();
    void destroy();
    void stack_count();
    void create();
  int count = 0;
  void main().
  {
      int no, ch, e;
  printf("1. push \n  2. pop \n  3. top \n
   4. Empty \n  5. Exit \n  6. Display \n
  7. Stack count \n  8. Destroy stack \n ");
  create();
 while(1)
  {
      printf("\n Enter choice : ");
       scanf("%d", &ch);
    switch(ch) {.
```

```
case 1 :
    printf ("Enter data : ");
    scanf ("%d", &no);
    push (no);
    break;
case 2 :
    pop();
    break;
case 3 :
    if (top == NULL)
      printf ("No elements in stack");
    else
     {
       e = topelement();
       printf ("\n Top.element : %d ",e);
     }
    break;
case 4 :
    empty();
     break;
case 5 :
    exit (0);
case 6 :
     display()
    break;
case 7 :
     stack_count();
     break;
case 8 :
    destroy();
    break;
```

```
        default :
            printf (" Wrong choice , Please enter
                        correct choice ");
            break;
        }
    }
}
/* Create Empty stack */

    void create() {
        top = NULL;
    }
/* Count stack elements */

    void stack_count () {
        printf (" No. of elements in stack : %d", count);
    }

/* Push data into stack */

    void push (int data)
    {
        if (top == NULL)
        {
            top = (struct node *) malloc ( 1* sizeof (struct node));
            top -> ptr = NULL;
            top -> info = data;
        }
    else
    {
        temp : (structmode *) malloc ( 1* size of (structnode));
        temp -> ptr = top;
        temp -> info = data ;
        top = temp; }   count ++;   }.
```

```
/* Displaying stack elements */
void display ()
{
    top1 = top;
    if (top1 == NULL)
    {
      printf ("stack is empty");
      return;
    }
printf ("stack elements are: \n");
while (top1 != NULL)
{
    printf ("%d", top -> info);
    top1 = top1 -> ptr;
}
}

/* pop operation on stack */

void pop()
{
    top1 = top;
    if (top1 == NULL)
    {
      printf (" Error: Trying to pop from
                empty stack ");
      return;
    }
else
    top1 = top1 -> ptr;
printf (" popped value %d ", top -> info);
free (top);
top = top1;
count--;  }
```

```
/* Return top element */
int topelement()
{
  return (top->info);
}
// check empty or not
void empty()
{
  if (top==NULL)
    printf ("\n stack is empty ");
  else
    {printf (" stack not empty with %d elts",count);
    }
}
/* Destroy entire stack */
void destroy()
  {
    top1 = top;
    while (top1!=NULL)
    {
      top1 = top->ptr;
      free(top);
      top = top1;
      top1 = top1->ptr;
    }
  free(top1);
  top=NULL;
  printf ("\n All stack elts destroyed");
  count=0;
}
```

OUTPUT:

```
1. push
2. pop
3. TOP
4. Empty
5. Exit
6. Display
7. Stack count
8. Destroy stack

Enter choice : 4.
    Stack is empty.

Enter choice : 1
 Enter data : 3.

Enter choice : 1
 Enter data : 6

 Enter choice : 1
  Enter data : 9

Enter choice : 6
   Stack elements are
    7  6  3

 Enter choice : 7
No. of elements in stack: 3.

 Enter choice : 5
```

WAP to demonstrate Queue Data structures using linked lists.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
  int info;
  struct node *ptr;
} *front, *rear, *temp, *front1;

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();
    int count = 0;

void main()
  {
    int no, ch, e;

  printf("\n 1. Enqueue \t 2. Dequeue \t
3. Front element \t 4 Empty \n 5. Exit \t 6. Display \t
7. Queue size \n");

    create()
      while(1) {
```

```
{
  printf ("\n Enter choice : ");
  scanf ("%d", &ch);
switch(ch)
 {
    case 1:
        printf ("Enter data %d", no);
        scanf ("%d", &no);
        enq(no);
        break;

    case 2:
          deq();
          break;

    case 3:
          e = frontelement();
          if (e!=0)
              printf ("Front element : %d", e);
          else
              printf ("\n No front element in
           Queue as queue is empty");
          break;
      case 4:
          empty();
            break;
      case 5:
            exit(0);

       case 6:
            display();
            break;
      case 7:
            queuesize();
            break;
```

```
        default :
            printf ("wrong choice");
        break;
        }
    }
}

/* create an empty queue */
  void create()
  {
    front = rear = NULL;
  }
/* returns Queue size */
   void queuesize ()
   {
     printf ("\n Queue size : %d ", count);
   }
/* Enqueing  Enqueueing the queue */
   void enq (int data)
      {
        if (rear == NULL)
        { rear = (struct node *)malloc (1* sizeof
                                        (struct node));
          rear -> ptr = NULL;
          rear -> info = data;
          front = rear;
        }
      else.
        {
          temp = (struct node*) malloc ( 1* sizeof (struct node));
          rear -> ptr = temp;
             temp -> info = data;  temp -> ptr = NULL;
```

```
        rear = temp;
        }
      count++;
    }
/* Displaying the queue elts */
    void display()
    {
      front1 = front;
      if ((front1 == NULL) && (rear == NULL))
      {
        printf ("Queue is empty");
        return;
      }
      while (front1 != rear)
      {
        printf ("%d", front1 -> info);
        front1 = front1 -> ptr;
      }
      if (front1 == rear)
        printf ("%d", front1 -> info);
    }
/* Dequeuing the queue */
    void deq()
    {
      front1 = front;
      if (front1 == NULL)
      {
        printf ("Error");
        return;
      }
      else
```

```
if (front1 -> ptr != NULL)
{
    front1 = front1 -> ptr;
    printf ("\n Dequeued value: %d", front->info);
    free (front);
    front = front1;
  }
else
  {
   printf ("\n Dequeued value : %d", front->info);
   free (front);
   front = NULL;
   rear = NULL;
  }
  count --;
}
/* returns front element in queue */
int frontelement()
 {
   if ((front != NULL) && (rear != NULL))
       return (front -> info);
    else
       return 0;
   }
/* Display if queue's empty / not */
void empty ()
 {
```

```
if (front == NULL) && (rear == NULL).
    printf ("\n Queue empty ");
else.
    printf ("Queue not empty ");
}
```

OUTPUT:

1. Enqueue 2. Dequeue 3. Front element 4 Empty
5 Exit 6. Display 7. Queue size.

Enter choice : 1
Enter data : 4.
Enter choice : 1
Enter data : 9
Enter choice : 1
Enter data : 17.
Enter choice : 1
Enter data : 69
Enter choice : 6
4 9 17 69
Enter choice : 3
Front element : 4
Enter choice : 4
Queue not empty
Enter choice : 7
Queue size : 5

WAP to demonstrate selection sort.

```
#include <stdio.h>
int main()
{
    int a[50], n, i, j, temp;
        printf ("Enter no. of elts : \n");
        scanf ("%d", &n);
        printf ("Enter %d integers \t", n);
                    one by one.

    for (i=0; i<n; i++){
            scanf ("%d", &a[i]);
        }

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

O/p: Enter no. of elts : 5
Enter all elements : 5 8 2 9 7
Sorted list : 2 5 7 8 9

WAP to demonstrate bubble sort.

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

O/P: Enter no. of elts : 5
Enter 5 integers : 3 8 2 9 5
Sorted bubble list : 2 3 5 8 9

WAP to demonstrate insertion sort.

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

O/P:

Enter no. of elts : 5

enter 5 integers : 1 8 3 9 4

Sorted insertion list : 1 3 4 8 9

WAP to demonstrate Quick sort.

```
#include <stdio.h>
void quicksort (int a[25], int first, int last)
 {
    int i, j, pivot, temp;
    if (first < last)
    {
      pivot = first;
       i = first;
      j = last;
       while (i<j)
      {
          while (a[i] <= a[pivot] && i<last)
          {
            i++; }
      while (a[j] > a[pivot]) {
          j--
        }
       if (i<j)
        {
          temp = a[i];   a[i] = a[j];   a[i] = temp;
        }
      }
    temp = a[pivot];
    a[pivot] = a[j];
     a[j] = temp;

    quicksort (a, first, j-1);
    quicksort (a, j+1, last);

  }
}

  int main ()
  {
   int i, n, a[25];
```

```
printf ("Enter total no. of elts: \n");
scanf ("%d", &n);
printf ("Enter the elements: \n");
for (i=0; i<n; i++)
  {
    scanf ("%d", &a[i]);
  }
quicksort (a, 0, n-1);
printf ("Sorted Array: \n");
for (i=0; i<n; i++)
  {
    printf ("%d \t", a[i]);
  } return 0;
}
```

O/p:

```
Enter total no. of elts : 5
Enter the elements
8
3
9
4
2
Sorted Array :
2 3 4 8 9
```

WAP to demonstrate Merge sort.

```
#include <stdio.h>
# define max 10

int a[11] = { 45, 23, 12, 44, 35, 61, 60, 21, 36, 62, 4};
int b[10];
void merging (int low, int mid, int high)
{
  int l1, l2, I;
  for (l1 = low, l2 = mid+1, I = low, l1<=mid &&
  l2<=high; I++)
{
  if (a[l1] <= a[l2])
     a[I] = a[l1++];
   else
      b[I] = a[l2++];
  }
while (l1 <= mid)
     b[I++] = a[l1++];
while (l2 <= high)
    b[I++] = a[l2++]
     for (I = low; I <= high; I++)
    a[I] = b[I];
  }
void sort (int low, int high)
{
  int mid;
    if (low < high)
{
   mid = (low + high) / 2;
```

```
    sort (low, mid);
     sort (mid+1, high);
    merging (low, mid, high);
  }
   else
     return;
 }
void main ()
 {
   int I ;
     printf (" List before sorting \n");
     for (I=0 ; I<= max ; I++)
     {
       printf (" %d" ;a[I]);
     }
   sort (0, max);
    printf (" \n List after sorting \n ");
    for (I = 0 ; I <= max ; I++)
    {
      printf (" %d \t", a[I]);
    }
 }
```

OUTPUT:

List befor sorting:

45 23 12 44 35 61 60 21 36 62 4

List after sorting:

4 12 21 23 35 36 44 45 60 61 62

WAP to read two sets and 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 ("%d", &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 A elements : 5 10 15 120 25 30
A intersection B is : 15 120

WAP to read two sets and 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 ("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; */
    C[i] = A[i];
  }
  k=m;
   for (i=0; i<n; i++){
          for (j=0; j<m; j++)
              if (B[i] == A[j])
                  break;
          if (j>=m)
              C[k++] = 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.
```

WAP to find m power n (m^n)

```
#include <stdio.h>
int power (int m, int n) {
        int i, p;
for (p. 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.

WAP 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 start & end (start< end : \n");
scanf ("%d %d", &start, &end );
printf ("perfect nos. b/w %d & %d are :\n", start,
                                        end);

for (num = start ; num <= end ; num++)
        if (isPerfect (num))
            printf ("%d", num);

}
```

OUTPUT :

Enter start & end (start< end): 1 10000

6 28 496 8128

WAP to find GCD of 4 numbers 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 numbers : \n");
        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 four numbers : 45 54 90 999

gcd of 45, 54, 90, 999 is 9.

WAP for area and 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 and breadth of rectangle:");
    scanf ("%d %d", &l, &b);
    Rect ( l, b, &ar, &pr);
    printf ("Area is %d \n Perimeter : %d \n", ar, pr);
}
```

OUTPUT:

Enter length and breadth of rectangle : 9 7

Area is : 63

perimeter is : 32

WAP 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 first, last, n;
    printf("enter first & last values in range: ");
    scanf("%d %d", &first, &last);
    printf("Prime nos. b/w %d & %d are: \n", first, last);
    for (n = first; n <= last; n++) {
        if (isPrime(n)) {
            printf("%d", n);
        }
    } printf("\n");
}
```

OUTPUT:

Enter first & last values in range: 67 150

prime nos. b/w 67 and 150 are: 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 139, 149,