TABLE OF CONTENTS

1.Find largest out of three numbers:

Algorithm:

Name: largest out of three numbers

Input: a,b,c

Output: max

Steps:

STEP1: Start.

STEP2: Input a,b,c.

STEP3: If a>b then

  STEP3.1: max=a.

  STEP3.2: End if

  STEP3.3: Else

  STEP3.4: max=b.

  STEP3.5: End if

STEP4: If max>c then max=c.

STEP5: End if

STEP6: Print max.

STEP7: End.

Program:

```
#include<stdio.h>
int main()
{
        int a,b,c,max;
        printf("Enter three numbers: ");
        scanf("%d%d%d",&a,&b,&c);
        if(a>b)
        max=a;
        else
        max=b;
        if(max<c)
        max=c;
printf("Biggest number is: %d",max);
}
```

Output:

Enter three numbers: 9 7 11
Biggest number is: 11

2.Centigrade to Fahrenheit:

Algorithm:

Name: Centigrade to Fahrenheit

Input: c

Output: f

Steps:

STEP1: Start.

STEP2: Input c.

STEP3: f=(9*c+160)/5.

STEP4: print f.

STEP5: Stop.

Program:

```
#include<stdio.h>
int main()
{
        float c,f;
        printf("enter celcius temperature ");
        scanf("%f",&c);
        f=(9*c+160)/5;
        printf("fahrenheit temperature is %f",f);
}
```

Output:

enter celcius temparature 28
fahrenheit temparature is 82.400002

3.Interchange between two numbers:

With taking third variable:

Algorithm:

Name: Interchange between two numbers.

Input: a,b

Output: a,b

Steps:

STEP1: Start.

STEP2: Input.

STEP3: c=a.

STEP4: a=b.

STEP5: b=c.

STEP6: Print a,b.

STEP7: End.

Program:

```
#include<stdio.h>
 int main()
{
   int a,b,c;
   printf("Enter first no: ");
   scanf("%d",&a);
   printf("Enter second no: ");
   scanf("%d",&b);
   c=a;
   a=b;
   b=c;
   printf("First no: %d",a);
   printf("\nSecond no: %d",b);
}
```

Output:

Enter first no: 7

Enter second no: 10

First no: 10

Second no: 7

Without taking third variable:

Algorithm:

Name: Interchange between two numbers.

Input: a,b

Output: a,b

Steps:

STEP1: Start.

STEP2: Input.

STEP3: a=a+b.

STEP4: b=a-b.

STEP5: a=a-b.

STEP6: Print a,b.

STEP7: End.

Program:

```
#include<stdio.h>
 int main()
{
   int a,b,c;
   printf("Enter first no: ");
   scanf("%d",&a);
   printf("Enter second no: ");
   scanf("%d",&b);
   a=a+b;
   b=a-b;
   a=a-b;
   printf("First no: %d",a);
   printf("\nSecond no: %d",b);
}
```

Output:

Enter first no: 7

Enter second no: 10

First no: 10

Second no: 7

4.Grade of a student taking percentage of marks as input:

Algorithm:

Name: Grade of a student.

Input: pmark.

Output:

Steps:

STEP1: Start.

STEP2: Input pmark.

STEP3: If pmark>=80 then go to STEP 3.1, else go to STEP4.

STEP3.1: Print Grade A.

Step3.2: End.

STEP4: If pmark>=60 then go to STEP4.1, else go to STEP5.

STEP4.1: Print Grade B.

STEP4.2: End if

STEP5: If pmark<=40 then go to STEP 5.1, else go to STEP6.

STEP5.1: Print Grade C.

Step5.2: End if

STEP6: If pmark<40.

STEP6.1: Print Grade D.

STEP6.2: End if

STEP7: End

Program:

```
#include<stdio.h>
int main()
{
        float pmark;
        printf("enter percentage of marks ");
        scanf("%f",&pmark);
        if(pmark>=80)
        printf("Grade A");
        else if(pmark>=60)
        printf("Grade B");
        else if(pmark>=40)
        printf("Grade C");
        else
        printf("Grade D");
}
```

Output:

```
enter percentage of marks 65
Grade B
```

5.Mobile Bill calculation for different unit price:

Algorithm:

Name: Mobile bill calculation.

Input: unit.

Output: bill.

Steps:

STEP1: Start.

STEP2: Input unit.

STEP3: If unit<=200 then

STEP3.1: bill=0.0 then go to STEP8.

STEP3.2: End if

STEP3.3: Else

STEP4: If unit<=300 then

STEP4.1: bill=5*(unit-200) then go to STEP8.

STEP4.2: End if

STEP4.3: Else

STEP5: If unit<=400 then

STEP5.1: bill=10000+6*(unit-300) then go to STEP8.

STEP5.2: End if

STEP5.3: Else

STEP6: If unit<=500 then

STEP6.1: bill=28000+7*(unit-400) then go to STEP8.

STEP6.2: End if

STEP6.3: Else

STEP7: bill=56000+10*(unit-500).

STEP8: End if

STEP8: Print bill.

STEP9: End.

Program:

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

```
 int main()
{
   int unit;
   float bill;
   printf("Enter unit: ");
   scanf("%d",&unit);
   if(unit<=200)
          bill=0.0;
   else if(unit<=300)
          bill=5*(unit-200);
   else if(unit<=400)
          bill=10000+6*(unit-300);
   else if(unit<=500)
          bill=28000+7*(unit-400);
   else
          bill=56000+10*(unit-500);
   printf("Bill: %0.2f",bill);
}
```

Output:

```
Enter unit: 500
Bill: 28700.00
```

6.Electric Bill calculation of WBSCDL for domestic urban connection:

Algorithm:

Name: Electric bill calculation.

Input: unit, F.

Output: charge.

Steps:

STEP1: Start.

STEP2: Input unit,F.

STEP3: If unit<=102 then

STEP3.1: charge=unit*5.04, then go to STEP8.

STEP3.2: End if

STEP3.3: Else

STEP4: If unit<=180 then

STEP4.1: charge=514.08+(unit-102)*6.33, then go to STEP8.

STEP4.2: End if

STEP4.3: Else

STEP5: If unit<=300 then

STEP5.1: charge=1007.82+(unit-180)*7.12, then go to STEP8.

STEP5.2: End if

STEP5.3: Else

STEP6: If unit<=600 then

STEP6.1: charge=1862.22+(unit-300)*7.52, then go to STEP8.

STEP6.2: End if

STEP6.3: Else

STEP7: charge=4118.22+(unit-600)*7.69.

STEP7.1: End if

STEP8: Print F*30

STEP9: Print charge.

STEP10: End.

Program:

```
#include<stdio.h>
int main()
{
        float unit,f,charge;
        printf("enter total power consumed in KWH");
        scanf("%f",&unit);
        printf("enter connected load in KVA");
        scanf("%f",&f);
        if(unit<=102)
        charge=5.04*unit;
        else if(unit<=180)
        charge=514.08+6.33*(unit-102);
        else if(unit<=300)
        charge=1007.82+7.12*(unit-180);
        else if(unit<=600)
        charge=1862.22+7.52*(unit-300);
        else
        charge=4118.22+7.69*(unit-400);
        printf("total fixed charges amount=Rs %0.2f",f*30);
        printf("\ntotal bill amount=Rs %0.2f",charge);
}
```

Output:

```
enter total power consumed in KWH 375
enter connected load in KVA 10
total fixed charges amount=Rs 300.00
total bill amount=Rs 2426.22
```

7.Test whether a year is leap year or not:

Algorithm:

Name: Checking a year is leap year or not.

Input: year

Output:

Steps:

STEP1: Start.

STEP2: If year%100=0 then go to STEP2.1, else go to STEP3.

STEP2.1: If year%400=0 then go to STEP4, else go to STEP5.

STEP3: If year%4=0 then go to STEP4, else go to STEP5.

STEP4: Print “Leap year”.

STEP5: Print “Not leap year”.

STEP6: End.

Program:

```
#include<stdio.h>
int main()
{
        int y;
        printf("enter a year ");
        scanf("%d",&y);
        if(y%100==0)
        {
                if(y%400==0)
                printf("leap year ");
                else
                printf("not leap year ");

        }
        else
        {
                if(y%4==0)
                printf("leap year ");
                else
                printf("not leap year ");
        }
}
```

Output:

```
enter a year 2016
leap year
```

8.Find day of the week of a date in 2023 taking 1st January,2023 as reference date:

Algorithm:

Name:
Input: d,m,y
Output:
Steps:

Step1: Start
Step2: Input d,m,y
Step3: diff=0
Step4: If(m-1)<=11 then diff=diff+30
Step5: If(m-1)<=10 then diff=diff+31
Step6: If(m-1)<=9 then diff=diff+30
Step7: if(m-1)<=8 then diff=diff+31
Step8: If(m-1)<=7 then diff=diff+31
Step9: If(m-1)<=6 then diff=diff+30
Step10: If(m-1)<=5 then diff=diff+31
Step11: If(m-1)<=4 then diff=diff+30
Step12: If(m-1)<=3 then diff=diff+31
Step13: If(m-1)<=2 then diff=diff+28
Step14: If(m-1)=1 then diff=diff+31
Step15: End if
Step16: diff(diff+d-1)
Step17: dow=diff%7
Step18: If dow=0 then print "sun" then go to Step26
Step19: If dow=1 then print "mon" then go to Step26
Step20: If dow=2 then print "tue" then go to Step26
Step21: If dow=3 then print "wed" then go to Step26
Step22: If dow=4 then print "thu" then go to Step26
Step23: If dow=5 then print "fri" then go to Step26
Step24: If dow=6 then print "sat" then go to Step26
Step25: End if
Step26: End

Program:

```
#include<stdio.h>
int main()
{
int d,m,y,diff,dow;
printf("enter a date of year 2023 in dd/mm/yyyy format");
scanf("%d%d%d",&d,&m,&y);
switch(m-1)
{
case 11:diff=diff+30;
case 10:diff=diff+31;
```

```
case 9:diff=diff+30;
case 8:diff=diff+31;
case 7:diff=diff+31;
case 6:diff=diff+30;
case 5:diff=diff+31;
case 4:diff=diff+30;
case 3:diff=diff+31;
case 2:diff=diff+28;
case 1:diff=diff+31;
}
diff=diff+d-1;
dow=diff%7;
switch(dow)
{
case 0:printf("sunday");break;
case 1:printf("monday");break;
case 2:printf("tuesday");break;
case 3:printf("wednesday");break;
case 4:printf("thursday");break;
case 5:printf("friday");break;
case 6:printf("saturday");break;
}
}
```

Output:

```
enter a date of year 2023 in dd/mm/yyyy format 10 10 2023
Tuesday
```

9.Find sum of digits of a number:

Algorithm:

Name: sum of digits of a number.

Input: n

Output: s

Steps:

STEP1: Start

STEP2: Input n

STEP3: s=0.

STEP4: If n!=0 then go to STEP4.1, else go to STEP5

STEP4.1: m=n%10.

STEP4.2: s=s+m.

STEP4.3: n=n/10 then go to STEP4.

STEP5: Print s.

STEP6: End.

Program:

```
#include<stdio.h>
int main()
{
        int n,s,m;
        printf("enter a number ");
        scanf("%d",&n);
        s=0;
        while(n!=0)
        {
                m=n%10;
                s=s+m;
                n=n/10;
        }
        printf("sum of digits %d ",s);
}
```

Output:

```
enter a number 357
sum of digits 15
```

10.Convert a decimal no into binary no:

Algorithm:

Name: Decimal to Binary.
Input: dec.
Output: s
Steps:

STEP1: Start.

STEP2: Input dec.

STEP3: s=0

STEP4:m=1

STEP5: If dec!=0 then go to STEP5.1, else go to STEP6

STEP5.1: r=dec/2;

STEP5.2: s=s+r*m

STEP5.3: m=m*10

STEP5.4: dec=dec/2 then go to STEP5

STEP6: Print s.

STEP7: End.

Program:

```
#include<stdio.h>
int main()
{
        int dec,s,m,r;
        printf("enter a number ");
        scanf("%d",&dec);
        s=0;
        m=1;
        while(dec!=0)
        {
                r=dec%2;
                s=s+r*m;
                m=m*10;
                dec=dec/2;
        }
        printf("result is=%d",s);
}
```

Output:

Enter a decimal value: 11
In binary form: 1011

11.Calculate factors of a number:

Algorithm:

Name: Factors of a number.

Input: n

Output: r

Steps:

STEP1: Start.

STEP2: Input n.

STEP3: d=1.

STEP4: If d<=n then go to STEP4.1, else go to STEP6

STEP4.1: r=n%d

STEP4.2: If r=0 then go

STEP4.2.1: Print d

STEP4.2.2: End if

STEP4.2.3: Else

STEP5: d=d+1 then go to STEP4

STEP6: End.

Program:

```
#include<stdio.h>
int main()
{
        int n,d,r;
        printf("enter a number: ");
        scanf("%d",&n);
        d=1;
        printf("Factors of this number: ");
        while(d<=n)
        {
                r=n%d;
                if(r==0)
                printf("\n%d",d);
                d++;
        }
}
```

Output:

enter a number 24

1
2
3
4
6
8
12
24

12.Convert a binary no into decimal no:

Algorithm:

Name: Binary to Decimal

Input: b

Output: s

Steps:

STEP1: Start

STEP2: Input b

STEP3: s=0

STEP4: count=0

STEP5: If b>0 then go to STEP5.1, else go to STEP6.

STEP5.1: d=b%10

STEP5.2: b=b/10

STEP5.3: s=s+d*pow(2,count)

STEP5.4: count=count+1 then go to STEP5.

STEP6: Print s

STEP7: End.

Program:

```
#include<stdio.h>
int main()
{
int s=0,d,count;
long int b;
printf("enter a binary number  ");
scanf("%ld",&b);
```

```
for(count=0;b>0;count++)
{
d=b%10;
b=b/10;
s=s+d*pow(2,count);
}
printf("resu;t is %d",s);
}
```

Output:

```
enter a binary number 1010
 result is 10
```

13.Prime numbers in between 1 to 1000:

Algorithm:

Name: Prime numbers between 1-1000

Input:

Output: i

Steps:

STEP1: Start.

STEP2: count=0

STEP3 i=1

STEP4: If i<=1000 then go to STEP4.1, else go to STEP5

  STEP4.1: d=1

  STEP4.2: If d<=i then go to STEP4.2.1, else go to STEP4.3

    STEP4.2.1: r=i%d.

    STEP4.2.2: If r=0 then count=count+1 and d=d+1 then go to STEP4.2

  STEP4.3: If count=2 then Print i

  STEP4.4: End if

  STEP4.5: count=0

  STEP4.6: i=i+1 then go to STEP4

STEP5: End

Program:

```
#include<stdio.h>
int main()
{
        int i,d,count;
        printf("Prime numbers in between 1 to 1000 are:- ");
   for(i=1;i<=1000;i++)
   {
        count=0;
        for(d=1;d<=i;d++)
        {
            r=i%d;
                if(r==0)
                 count++;
                 }
                 if(count==2)
                 printf("%6d",i);
        }
}
```

Output:

```
Prime numbers in between 1 to 1000 are:-     2    3    5    7   11   13   17   19   23   29   31   37
41   43   47   53   59   61   67   71   73   79   83   89   97  101  103  107  109  113  127  131
137  139  149  151  157  163  167  173  179  181  191  193  197  199  211  223  227  229
233  239  241  251  257  263  269  271  277  281  283  293  307  311  313  317  331  337
347  349  353  359  367  373  379  383  389  397  401  409  419  421  431  433  439  443
449  457  461  463  467  479  487  491  499  503  509  521  523  541  547  557  563  569
571  577  587  593  599  601  607  613  617  619  631  641  643  647  653  659  661  673
677  683  691  701  709  719  727  733  739  743  751  757  761  769  773  787  797  809
811  821  823  827  829  839  853  857  859  863  877  881  883  887  907  911  919  929
937  941  947  953  967  971  977  983  991  997
```

14.Perfect number in between 1 to 1000:

Algorithm:

Name: Perfect numbers in between 1-1000

Input:

Output: i

Steps:

STEP1: Start

STEP2: s=0

STEP3: i=1

STEP4: If i<=1000 then go to STEP4.1, else go to STEP5

STEP4.1: d=1

STEP4.2: If d<= i then go to STEP4.2.1, else go to STEP4.3

STEP4.2.1: r=i%d

STEP4.2.2: If r=0 then s=s+d and d=d+1 then go to STEP4.2

STEP4.3: If s=i then print i

STEP4.4: End if

STEP4.4: s=0

STEP4.5: i=i+1 then go to STEP4

STEP5: End

Program:

```
#include<stdio.h>
int main()
{
        int i,d,s;
        printf("Perfect numbers in between 1 to 1000 are:- ");
  for(i=1;i<=1000;i++)
  {
        count=0;
        for(d=1;d<i;d++)
        {
                r=i%d;
                if(r==0)
```

```
                s=s+d;
                }
                if(s==i)
                printf("\n%d",i);
        }
}
```

Output:

Prime numbers in between 1 to 1000 are:-
6
28
496

15.Armstrong number in between 1 to 1000:

Algorithm:

Name: Armstrong number between 1-1000

Input:

Output: copy1

Steps:

STEP1: Start

STEP2: n=1

STEP3: If n<=1000 then go to STEP3.1, else go to STEP4

STEP3.1: copy1=n

STEP3.2: count=0

STEP3.3: If n>0 then go to STEP 3.3.1, else go to STEP3.4

STEP3.3.1: n=n/10

STEP3.3.2: count=count+1 then go to STEP3.3

STEP3.4: n=copy1

STEP3.5: s=0

STEP3.6: If n>0 then go to STEP3.6., else go to STEP3.7

STEP3.6.1: digit=n%10

STEP3.6.2: s=s+pow(digit,count)

STEP3.6.3: n=n/10 them go to STEP3.6

STEP3.7: If copy1=s then print copy1

STEP3.8: End if

STEP3.9: n=copy1

STEP3.10: count=0

STEP3.11: n=n+1 then go to STEP3

STEP4: End

Program:

```
#include<stdio.h>
#include<math.h>
int main()
{
   int count,s,n,copy1,digit;
   printf("Armstrong numbers in between1 to 1000 are:-  ");
   for(n=1; n<=1000; n++)
   {
      copy1=n;
      for(count=0; n>0; count++)
      {
         n=n/10;
      }
      n=copy1;
      for(s=0; n>0; n=n/10 )
      {
         digit=n%10;
         s=s+pow(digit,count);
      }
      if(copy1==s)
         printf("%-6d",copy1);
      n=copy1;
      count=0;
   }
}
```

Output:

Armstrong numbers in between 1 to 1000 are:- 1 2 3 4 5 6 7 8 9 153 370 371 407

16.Nearest Prime of an input number:

Algorithm:

Name: Nearest prime of a number.

Input: n

Output:
Steps:

STEP1: Start

STEP2: Input n

STEP3: F=n

STEP4: n=n-1

STEP5: If n>1 then go to STEP5.1, else go to STEP6

STEP5.1: d=2

STEP5.2: If d<=n/2 then go to STEP5.2.1, else go to STEP5.3

STEP5.2.1: r=n%d

STEP5.2.1: If r=0 then go to STEP5.3, else go to STEP5.2.1.1

STEP5.2.1.1: d=d+1 then go to STEP5.2

STEP5.3: save=n

STEP5.4: n=n-1

STEP5.5: If r!=0 then go to STEP6, else go to STEP5

STEP6: n=F

STEP7: n=n+1

STEP8: If n>1 then go to STEP8.1, else go to STEP9

STEP8.1: d=2

STEP8.2: If d<=n/2 then go to STEP8.2.1, else go to STEP8.3

STEP8.2.1: r=n%d

STEP8.3.1: If r=0 then go to STEP8.3, else go to STEP38.3.1.1

STEP8.3.1.1: d=d+1 then go to STEP8.2

STEP8.3: SAVE=n

STEP8.4: n=n-1

STEP8.5: If r!=0 then go to STEP9, else go to STEP8

STEP9: If F=2 then print “nearest prime number 3”

STEP10: End if

STEP11: else

STEP10: If F-save=SAVE-F then print save, SAVE

STEP12: End if

STEP13: else

STEP11: If F-save<SAVE-F then print save, else print SAVE

STEP14: End if

STEP15: End

Program:

```
#include<stdio.h>
int main()
{
int r,d,n,save,SAVE,F;
printf("Enter a number:  ");
scanf("%d",&n);
F=n;
n=n-1;
while(n>1)
  {
    for(d=2;d<=n/2;d++)
    {
      r=n%d;
      if(r==0)
      break;
    }
  save=n;
  n--;
  if(r!=0)
  break;
  }
  n=F;
n=n+1;
while(n>1)
  {
    for(d=2;d<=n/2;d++)
    {
      r=n%d;
      if(r==0)
      break;
    }
  SAVE=n;
  n++;
```

```
  if(r!=0)
  break;
  }
 if(F==2)
  printf("nearest prime no: 3");
else if(F-save==SAVE-F)
  printf("Nearest prime numbers are both %d and %d",save,SAVE);
else
  (F-save<SAVE-F)?printf("nearest prime no: %d",save):printf("nearest prime no: %d",SAVE);
}
```

Output:

```
enter a number 50
nearest prime numbers are both 53 and 47
```

17.Calculate HCF and LCM of two numbers:

Algorithm:

Name: HCF and LCM of two numbers

Input: a,b

Output: HCF,LCM

Steps:

STEP1: Start

STEP2: Input a,b

STEP3: c=a*b

STEP4: If b%a!=0 then go to STEP4.1, else go to STEP5

STEP4.1: r=b%a

STEP4.2: b=a

STEP4.3: a=r then go to STEP4

STEP5: HCF=a

STEP6: LCM=c/a

STEP7: Print HCF

STEP8: Print LCM

STEP9: End

Program:

```
#include<stdio.h>
#include<math.h>
int main()
{
int a,b,r,c,HCF,LCM;
printf("enter two numbers");
scanf("%d%d",&a,&b);
c=a*b;
while(b%a!=0)
{
r=b%a;
b=a;
a=r;
}
HCF=a;
LCM=c/a;
printf("HCF is %d", HCF);
printf("\nLCM is %d", LCM);
}
```

Output:

```
enter two numbers 15 25
HCF is 5
LCM is 75
```

18.Find largest and smallest out of n numbers:

Algorithm:

Name: largest and smallest out of n numbers

Input: n

Output: max,min

Steps:

STEP1: start

STEP2: input n

STEP3: i=0

STEP4: if i<n then go to step 4.1 else go to step 5

STEP4.1: input a[i]

STEP4.2: i=i+1 then go to step 4.

STEP5: End if.

STEP6: max=a[0].

STEP7: min=a[0].

STEP8: i=0.

STEP9: if i<n then go to step 9.1 else go to step 10.

STEP9.1: if a[i]>max then go to step 9.2.

STEP9.2: max=a[i].

STEP9.3: if a[i]<min then go to step 9.4.

STEP9.4: min=a[i].

STEP9.5: i=i+1 then go to step 9.

STEP10: End if.

STEP11: Print max,min

Step12: End

Program:

```
#include<stdio.h>
int main()
{
int a[1000],n,i,max,min;
```

```
printf("enter number of elements");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter element");
scanf("%d",&a[i]);
}
max=a[0];
min=a[0];
for(i=0;i<n;i++)
{
if(a[i]>max)
max=a[i];
if(a[i]<min)
min=a[i];
}
printf("largest number=%d",max);
printf("\nsmallest number=%d",min);
}
```

Output:

```
enter number of elements in list 5
enter element 38
enter element 57
enter element 31
enter element 42
enter element 87
largest number= 87
smallest number= 31
```

19.Calculate prime factors of a number:

Algorithm:

Name: prime factors of a number

Input: n

Output: d

Steps:

STEP1: Start

STEP2: Input n

STEP3: d=2

STEP4: If n>1 then go to STEP4.1, else go to STEP5

STEP4.1: r=n%d

STEP4.2: If r=0 then go to STEP4.2.1, else go to STEP4.3

STEP4.2.1: Print d

STEP4.2.2: n=n/d

STEP4.2.3: End if

STEP4.3: d=d+1 then go to STEP4

STEP5: End

Program:

```
#include<stdio.h>
#include<math.h>
int main()
{
int n,d=2,r;
printf("enter a number");
scanf("%d",&n);
while(n>1)
{
r=n%d;
if(r==0)
{
printf("\n%d",d);
n=n/d;
}
else
d++;
```

```
}
}
```

Output:

```
enter a number 24

2
2
2
3
```

20.Make fibonacci sequence upto n terms:

Algorithm:

Name: Start

Input: n

Output: c

Steps:

STEP1: Start

STEP2: Input n

STEP3: a=0

STEP4: b=1

STEP5: Print a

STEP6: i=2

STEP7: If i<=n then go to STEP7.1, else go to STEP8

  STEP7.1: c=a+b

  STEP7.2: a=b

  STEP7.3: b=c

  STEP7.4: Print c

  STEP7.5: i=i-1 then go to STEP7

STEP8: End

Program:

```
#include<stdio.h>
int main()
{
   int a,b,c,i,n;
   printf("Enter total no of terms: ");
   scanf("%d",&n);
   a=0;
   b=1;
   printf("%-2d%2d",a,b);
   for(i=2; i<=n; i++)
   {
      c=a+b;
      a=b;
      b=c;
      printf("%4d",c);
   }
}
```

Output:

```
Enter total no of terms: 10
0  1   1   2   3   5   8  13  21  34  55
```

21.Find twin primes between 1-100:

Algorithm:

Name: Twin primes between 1-100

Input:

Output: i, i+2

Steps:

STEP1: Start.

STEP2: i=1

STEP3: If i<=100 then go to STEP3.1, else go to STEP4

STEP3.1: p1=0

STEP3.2: p2=0

STEP3.3: j=1

STEP3.4: If j<=i then go to STEP3.4.1, else go to STEP3.5

STEP3.4.1: If i%j=0 then go to STEP3.4.1.1

STEP3.4.1.1: p1=p1+1

STEP3.4.1.2: j=j+1 then go to STEP3.4

STEP3.5: j=1

STEP3.6: If j<=i+2 then go to STEP3.6.1, else go to STEP3.7

STEP3.6.1: If (i+2)%j=0 then go to STEP3.6.1.1

STEP3.6.1.1: p2=p2+1

STEP3.6.1.2: j=j+1 then go to STEP3.6

STEP3.7: If p1=2&&p2=2 then

STEP3.7.1: Print I,i+2

STEP3.7.2: End if

STEP3.7.3: i=i+1 then go to STEP3

STEP4: End

Program:

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

```
int main()
{
	int p1,p2,i,j;
	printf("Twin prime numbers between 1-100:\n");
	for(i=1;i<=100;i++)
	{
		p1=0;
		p2=0;
		for(j=1;j<=i;j++)
		{
			if(i%j==0)
			p1++;
		}
		for(j=1;j<=i+2;j++)
		{
			if((i+2)%j==0)
			p2++;
		}
		if(p1==2&&p2==2)
		printf("\n%d%3d",i,i+2);
	}
}
```

Output:

3 5

5 7

11 13

17 19

29 31

41 43

59 61

71 73

22.Make average out of n numbers:

Algorithm:

Name: Making average out of n numbers

Input: n, index number

Output: avg

Steps:

Step1: Input n

Step2: Repeat step3 for i=0 to n-1

If i>n Then go to step4

Step3: input digits, num[i] and then

i=i+1 and go to step2

Step4: Repeat step5 for i=0 to n-1

End if

If i>n Then go to step6

Else go to step7.

Step5: s=s+num[i] and then i=i+1

And go to step 4.

Step6: avg= s/n

Step7: print avg

End if

Step8: Stop.

Program:

```
#include<stdio.h>
int main()
{
        int num[100],i,n;
        float avg,s=0;
        printf(" Enter the index number ");
        scanf("%d",&n);
        for(i=0;i<=n-1;i++)
        {
                printf("Enter digits: ");
                scanf("%d",&num[i]);
        }

        for(i=0;i<n;i++)
        {
          s=s+num[i];
        }
        avg=s/n;
        printf(" Average number is %f... “, avg);
}
```

Output:

Enter the index number 4

Enter digits: 4

Enter digits: 6

Enter digits: 8

Enter digits: 14

Average number is 8.000000...

23.Linear search of 1D array:

Algorithm:

Name: Linear search

Input: arr[i],n

Output:

Steps:

Step1: Input number of elements in list,n

Step2: Repeat step3 for i=0 to n-1

If i>n Then go to step4

Step3: input digits, arr[i] and then

i=i+1 and go to step2

Step4: Input the key value to search (key)

End if

Step5: flag=0

Step6: Repeat step7 for i=0 to n-1

If n>i then go to step8

Step7: if key==arr[i] Then

Flag =1

m=i

i=i+1 and then go to step 6

Step8: if flag==1 Then

Print found at ,m+1

Else

Print Not found.

End if, End if

Step9: End

Program:

```
#include<stdio.h>
int main()
{
        int arr[100],n;
        int i,key,m,flag;
       printf(" Enter the number of elements in the list: ");
        scanf("%d",&n);
        for(i=0;i<=n-1;i++)
        {
                printf("Enter the elements: ");
                scanf("%d",&arr[i]);
        }
        printf("Enter the key value to search: ");
        scanf("%d",&key);
        flag=0;
        for(i=0;i<=n-1;i++)
        {
              if(key==arr[i])
                 {
                   flag=1;
                   m=i;
                 }
        }
        if (flag==1)
        printf("\nFound at position..%d",m+1);
        else
        printf("\n Not found ");

}
```

Output:

Enter the number of elements in the list: 8

Enter the elements: 9

Enter the elements: 21

Enter the elements: 65

Enter the elements: 32

Enter the elements: 97

Enter the elements: 56

Enter the elements: 45

Enter the elements: 75

Enter the key value to search45

Found at position.. 7

24.Bubble sort of 1D array:

Algorithm:

Name: Bubble sort

Input: arr[i],n

Output: arr[i],arr[j]

Step1: Input the elements in a array,n

Step2: Repeat step3 for i=0 to n-1

If i>n Then go to step4

Step3: input arr[i] and then

i=i+1 and go to step2

Step4: Repeat step5 for i=0 to n-1

If i>n Then go to step6

Step5: print arr[i]

i=i+1 and then go to step4

Step6: Repeat step 7 for pass=1 to n-1

If pass>n then go to step10

Step7: Repeat step8 for j=0 to n-pass-1

If j>n-pass-1 then go to step9

Step8: if arr[j]>arr[j+1] Then

temp= arr[j]

arr[j]=arr[j+1]

arr[j+1]=temp

j=j+1 and then go to step7

Step9: pass=pass+1 and go to step6

Step10: Repeat step11 for j=0 to n-1

If j>n then go to step 12

Step11: print arr[j]

j=j+1 and then go to step 10

Step12: Stop

Program:

```
#include<stdio.h>
int main()
{
        int arr[100];
        int i,j, pass, temp,n;
        printf("Enter the no of elements  in the arry");
        scanf("%d",&n);
        for(i=0;i<=n-1;i++)
        {
                printf("Enter elements: ");
                scanf("%d",&arr[i]);
        }
        printf(" print the array before sorting: ");
        for(i=0;i<=n-1;i++)
        {
                printf("%5d",arr[i]);
        }
        for(pass=1;pass<=n-1;pass++)
        {
                for(j=0;j<=n-pass-1;j++)
                {
                        if (arr[j]>arr[j+1])
                      {
                temp=arr[j];
                arr[j]=arr[j+1];
                        arr[j+1]=temp;
                        }
                }
        }
         printf("\nArray after sorting: ");
         for(j=0;j<=n-1;j++)
         {
                printf("%4d",arr[j]);
         }
}
```

Output:

Enter the no of elements in the array: 8

Enter elements: 95

Enter elements: 25

Enter elements: 65

Enter elements: 47

Enter elements: 23

Enter elements: 2

Enter elements: 59

Enter elements: 45

Given Output:

print the array before sorting 95 25 65 47 23 2 59 45

Array after sorting 2 23 25 45 47 59 65 95

25.Merging of two sorted arrays:

Algorithm:

Name: Merging of two sorted arrays

Input: arr[i],n

Output: arr[i]

Steps:

Step1: Input the elements in arr1,m

Step2: Repeat step3 for i=0 to m-1

If i>m Then go to step4

Step3: input arr1[i] and then

i=i+1 and go to step2

Step4: Input the elements in arr2,n

End if

Step5: Repeat step6 for j=0 to n-1

If j>n Then go to step7

Step6: input arr2[j] and then

j=j+1 and go to step5

Step7: End if

Repeat Step 8 for j=0 to m-1 And then

repeat step8 for i=0 to m-1

If j>m then go to step 9

if i>m then go to step 9

Step8: arr3[j]=arr1[i]

J= j+1 and then go to step 7

Step9: End if ,End if

Repeat step 10 for j=m to n-1and then

Repeat step 10 for i=0 to n-1

If I and j>n then go to step 11

Step10: arr3[j]=arr2[i]

j=j+1 and then go to step 9

Step11: End if, End if

Repeat step 12 for j=0 to m+n and also

If j>m+n then go to step 13

Step12: print arr3[j] then

J=j+1 and go to step 11

Step13: Stop.

Program:

```
#include<stdio.h>
int main()
{
 int i,j,m,n;
 int arr1[100],arr2[100],arr3[200];
 printf("Enter the size of 1st array and 2nd array: ");
 scanf("%d%d",&m,&n);
 for(i=0;i<m;i++)
 {
  printf("Enter elements of 1st array: ") ;
  scanf("%d",&arr1[i]);
 }
 for(i=0;i<n;i++)
 {
  printf("Enter elements of 2nd array: ") ;
  scanf("%d",&arr2[i]);
 }
 j=0;
 for(i=0;i<m;i++)
 {
  arr3[j]=arr1[i];
  j++;
 }
 for(i=0;i<n;i++)
 {
   arr3[j]=arr2[i];
   j++;
```

```
 }

 for(j=0;j<m+n;j++)
 {
  printf("%3d",arr3[j]);

 }

}
```

Output:

Enter the size of 1st array and 2nd array: 3 3

Enter elements of 1st array: 3

Enter elements of 1st array: 4

Enter elements of 1st array: 7

Enter elements of 2nd array: 9

Enter elements of 2nd array: 12

Enter elements of 2nd array: 15

3 4 7 9 12 15

26.Binary search:

Algorithm:

Name: binary search:

Input: arr[i],n,key

Output: mid

Steps:

Step1: Start

Step2: Input n

Step3: Repeat Step4 for i=0 to n-1

Step4: Input arr[i]

Step5: lb=0 and ub=n-1

Step6: input key

Step7: If lb<=ub then go to Step8, else go to Step

Step8: mid=lb+ub/2

Step9: If key=arr[mid] then

Step10: Print mid then go to Step16.

Step11: Else

Step12: If key>arr[mid] then

Step13: lb=mid+1

Step14: Else

Step15: ub=mid-1 then go to Step7

Step16: If lb>ub then

Step17: Print “Not Found”.

Step18: End if

Step19: End

Program:

```
#include<stdio.h>
int main()
{
        int arr[100],n,key,i,lb,ub,mid;
        printf("Enter no of elements ");
        scanf(" %d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter element ");
                scanf(" %d",&arr[i]);
        }
        lb=0;
        ub=n-1;
        printf("Enter key value ");
        scanf("%d",&key);
        while(lb<=ub)
    {
       mid=(lb+ub)/2;
          if(key==arr[mid])
           {
            printf("found at %d",mid+1);
            break;
           }
          else if(key>arr[mid])
            lb=mid+1;
          else
          ub=mid-1;
            }
        if(lb>ub)
        printf("Not found");
}
```

Output:

```
Enter no of elements 5
Enter element 10
Enter element 20
Enter element 30
Enter element 40
Enter element 50
Enter key value 20
found at 2
```

27.Find largest number of each row and columm in a matrix:

Algorithm:

Name: Maximun no in row and columm

Input: i,j,a[row][col]

Output: a[row][col]

Steps:

Step1: Start

Step2: Input i,j

Step3: row=0.

Step4: Repeat Step5 for row=0 to i-1

Step5: Repeat Step6 for col=0 to j-1

Step6: Input a[row][col]

Step7: Repeat Step8 for row=0 to i-1

Step7.1: max=a[row][0]

Step8: Repeat Step9 for col=0 to j-1

Step9: If max<a[row][col] then max=a[row][col], then go to Step9.1

Step9.1: Print max.

Step10: Repeat Step11 for row=0 to i-1

Step10.1: max=a[row][0]

Step11: Repeat Step12 for col=0 to j-1

Step12: If max<a[row][col] then max=a[row][col], then go to Step12.1

Step12.1: Print max.

Step13: End

Program:

```
#include<stdio.h>
int main()
{
   int a[100][100],max,row,col,i,j;
//taking matrix size>>
   printf("Enter matrix size:");
   scanf("%d %d",&i,&j);
```

```
//input taking>>
   for(row=0; row<=i-1; row++)
   {
      printf("Enter row %d\n",row+1);
      for(col=0; col<=j-1; col++)
      {
         printf("enter elements-->");
         scanf("%d",&a[row][col]);
      }
   }
//Maximum in row>>
   for(row=0; row<=i-1; row++)
   {
      max=a[row][0];
      for(col=0; col<=j-1; col++)
      {
         if(max<a[row][col])
            max=a[row][col];
      }
      printf("Maximum no in row %d-->%d\n",row+1,max);
   }
//Maximum in column>>
   for(col=0; col<=j-1; col++)
   {
      max=a[0][col];
      for(row=0; row<=i-1; row++)
      {
         if(max<a[row][col])
            max=a[row][col];
      }
      printf("Maximum no in column %d-->%d\n",col+1,max);
   }
}
```

Output:

```
enter elements-->3
Maximum no in row 1-->7
Maximum no in row 2-->6
Maximum no in column 1-->7
Maximum no in column 2-->2
Maximum no in column 3-->4
```

28.Find a matrix is symmetric or not:

Algorithm:

Name: Checking for a matrix is symmetric or not

Input: n, a[row][col]

Output:

Steps:

Step1: Start.

Step2: Input n

Step3: Repeat Step5 for row=0 to n-1

Step4: Repeat Step6 for col=0 to n-1

Step5: flag=0

Step6: Input a[row][col]

Step7: Repeat Step8 for row=0 to n-2

Step8: Repeat Step9 for col=row+1 to n-1

Step9: If a[row][col]!=a[col][row] then go to Step9.1.

Step9.1: flag=1;

Step9.2: If flag=1 then go to Step10

Step10: If flag=0 then print “Symmetric matrix”, else print ”not symmetric matrix”.

Step11: End

Program:

```
#include<stdio.h>
int main()
{
int a[10][10];
int row,col,n,flag=0;
printf("Enter matrix size: ");
scanf("%d",&n);
for(row=0;row<=n-1;row++)
{
printf("Enter row %d: \n",row+1);
for(col=0;col<=n-1;col++)
{
printf("enter elements-->");
```

```
scanf("%d",&a[row][col]);
}
}
for(row=0;row<=n-2;row++)
{
for(col=row+1;col<=n-1;col++)
{
if(a[row][col]!=a[col][row])
{
flag=1;
break;
        }
}
if(flag==1)
break;
}
if(flag==0)
printf("\n\nSymmetric matrix");
else
printf("\n\nnot a Symmetric matrix");
}
```

Output:

```
Enter matrix size: 3
Enter row 1:
enter elements-->1
enter elements-->1
enter elements-->-1
Enter row 2:
enter elements-->1
enter elements-->2
enter elements-->0
Enter row 3:
enter elements-->-1
enter elements-->0
enter elements-->-1
Symmetric matrix
```

29.Calculate trace of a matrix:

Algorithm:

Name: Calculate trace of a matrix

Input: n,a[row][col]

Output: s

Steps:

Step1: Start

Step2: Input n

Step3: s=0

Step4: Repeat Step5 for row=0 to n-1

Step5: Repeat Step6 for col=0 to n-1

Step6: Input a[row][col]

Step7: Repeat Step8 for row=0 to n-1

Step8: s=s+a[row][row]

Step9: Print s

Step10: End

Program:

```
#include<stdio.h>
int main()
{
int n ,a[100][100],row,col,s;
printf("Enter matrix size: ");
scanf("%d",&n);
for(row=0;row<=n-1;row++)
{
printf("Enter row no: %d",row+1);
for(col=0;col<=n-1;col++)
{
printf("\nEnter element>>: ");
scanf("%d",&a[row][col]);
}
}
for(row=0;row<=n-1;row++)
{
s=s+a[row][row];
}
```

```
printf("Trace of this matrix: %d",s);
}
```

Output:

```
Enter matrix size: 3

Enter row no: 1

Enter element>>: 1

Enter element>>: 2

Enter element>>: 3

Enter row no: 2

Enter element>>: 1

Enter element>>: 2

Enter element>>: 3

Enter row no: 3

Enter element>>: 1

Enter element>>: 2

Enter element>>: 3

Trace of this matrix: 6
```

30.Check a matrix is skew-symmetric or not:

Algorithm:

Name: Checking a program is skew-symmetric or not

Input: a[row][col],n

Output:

Steps:

Step1: Start

Step2: Input n

Step3: Repeat Step4 for row=0 to n-1

Step4: Repeat step5 for col=0 to n-1

Step5: Input a[row][col]

Step6: flag=0

Step7: Repeat Step8 for row=0 to n-2

Step8: Repeat Step9 for col=row+1 to n-1

Step9: if a[row][col]!=-a[col][row] then go to Step9.1

Step9.1: flag=flag+1

Step10: Repeat Step11 for row=0 to n-1

Step11: If a[row][row]!=0 then go to Step11.1

Step11.1: flag=flag+1

Step12: If flag!=0 then

Step12.2: Print “Skew-Symmetric matrix”

Step12.2: else

Step12.3: Print “Not a Skew-symmetric matrix”

Step12.4: End if

Step13: End

Program:

```
#include<stdio.h>
int main()
{
   int a[10][10];
```

```
   int row,col,n,flag=0;
   printf("Enter size of matrix: ");
   scanf("%d",&n);
   for(row=0; row<=n-1; row++)
   {
      printf("Enter row %d\n",row+1);
      for(col=0; col<=n-1; col++)
      {
         printf("enter elements-->");
         scanf("%d",&a[row][col]);
      }
   }
   flag=0;
   for(row=0; row<=n-2; row++)
   {
      for(col=row+1; col<=n-1; col++)
      {
         if(a[row][col]!=-a[col][row])
         {
            flag++;
            break;
         }
      }
      if(flag==1)
         break;
   }
   for(row=0; row<=n-1; row++)
   {
      if(a[row][row]!=0)
      {
         flag++;
         break;
      }
   }
   if(flag!=0)
      printf("\nNot a skew-symmetric matrix");
   else
      printf("\nSkew-Symmetric matrix");
}
```

Output:

Enter size of matrix: 3

Enter row 1

enter elements-->0

enter elements-->1

enter elements-->-2

Enter row 2

enter elements-->-1

enter elements-->0

enter elements-->3

Enter row 3

enter elements-->2

enter elements-->-3

enter elements-->0

Skew-Symmetric matrix

31.Make transpose of a matrix:

Algorithm:

Name: Transpose of a matrix

Input: a[row][col],i,j

Output: b[row][col]

Steps:

Step1: Start

Step2: Input i,j

Step3: Repeat Step4 for row=0 to i-1

Step4: Repeat step5 for col=0 to j-1

Step5: Input a[row][col]

Step6: Repeat Step7 for row=0 to i-1

Step7: Repeat step8 for col=0 to j-1

Step8: a[row][col]=b[col][row]

Step9: Repeat Step10 for row=0 to j-1

Step10: Repeat step11 for col=0 to i-1

Step11: Print b[row][col]

Step12: End

Program:

```
#include<stdio.h>
int main()
{
   int a[10][10],b[10][10];
```

```
   int i,j,row,col;
   printf("Enter elements of 1st matrix>>\n");
//input of row&columm no taking >>
   printf("enter total no of rows: ");
   scanf("%d",&i);
   printf("enter total no of columms: ");
   scanf("%d",&j);
//input taking>>
   for(row=0; row<=i-1;     row++;
  {
      printf("Enter row %d\n",row+1);
      for(col=0; col<=j-1; col++)
      {
         printf("enter elements>> ");
         scanf("%d",&a[row][col]);
      }
   }
   printf("\nTranspose of matrix-->\n");
   for(row=0; row<=i-1; row++)
   {
      for(col=0; col<=j-1; col++)
      {
         b[col][row]=a[row][col];
      }
   }
   for(row=0; row<=j-1; row++)
   {
      for(col=0; col<=i-1; col++)
      {
         printf("%3d",b[row][col]);
      }
      printf("\n");
   }
}
```

Output:

Enter elements of 1st matrix>>

enter total no of rows: 2

enter total no of columms: 3

Enter row 1

enter elements>> 1

enter elements>> 2

enter elements>> 3

Enter row 2

enter elements>> 4

enter elements>> 5

enter elements>> 6

Transpose of matrix-->

1 4

2 5

3 6

32.Make addition a subtraction of two matrices:

Algorithm:

Name: Addition and subtraction

Input: a[row][col], b[row][col],i,j

Output: c[row][col], d[row][col]

Steps:

Step1: Start

Step2: Input i,j

Step3: Repeat Step4 for row=0 to i-1

Step4: Repeat step5 for col=0 to j-1

Step5: Input a[row][col]

Step6: Repeat Step7 for row=0 to i-1

Step7: Repeat step8 for col=0 to j-1

Step8: Input b[row][col]

Step9: Repeat Step10 for row=0 to i-1

Step10: Repeat Step11and Step12 for col=0 to j-1

Step11: c[row][col]=a[row][col]+b[row][col]

Step12: d[row][col]=a[row][col]-b[row][col]

Step13: Repeat Step14 for row=0 to i-1

Step14: Repeat Step15 to col=0 to j-1

Step15: Print c[row][col]

Step16: Repeat Step17 for row=0 to i-1

Step17: Repeat Step18 for col=0 to j-1

Step18: Print d[row][col]

Step19: End

Program:

```
#include<stdio.h>
int main()
{
	int a[100][100],b[100][100],c[100][100],d[100][100];
	int i,j,row,col;
	printf("Enter no of rows and columns: ");
	scanf("%d%d",&i,&j);
	printf("Enetr elements of 1st matrix:\n");
	for(row=0;row<=i-1;row++)
	{
		printf("Enter elements of row number %d:\n",row+1);
		for(col=0;col<=j-1;col++)
		{
			printf("Enter elements>> ");
			scanf("%d",&a[row][col]);
		}
	}
	printf("Enetr elements of 2nd matrix:\n");
	for(row=0;row<=i-1;row++)
	{
		printf("Enter elements of row number %d:\n",row+1);
		for(col=0;col<=j-1;col++)
		{
			printf("Enter elements>> ");
			scanf("%d",&b[row][col]);
		}
	}
	printf("Addition of matrix:\n");
	for(row=0;row<=i-1;row++)
	{
	  for(col=0;col<=j-1;col++)
		{
			c[row][col]=a[row][col]+b[row][col];
			printf("%5d",c[row][col]);
		}
		printf("\n");
  }
  printf("subtraction of matrix:\n");
	for(row=0;row<=i-1;row++)
	{
	  for(col=0;col<=j-1;col++)
		{
			d[row][col]=a[row][col]-b[row][col];
			printf("%5d",d[row][col]);
		}
		printf("\n");
  }
}
```

Output:

```
Enter no of rows and columns: 2 3
Enetr elements of 1st matrix:
Enter elements of row number 1:
Enter elements>> 1
Enter elements>> 5
Enter elements>> 4
Enter elements of row number 2:
Enter elements>> 0
Enter elements>> 7
Enter elements>> 3
Enetr elements of 2nd matrix:
Enter elements of row number 1:
Enter elements>> 1
Enter elements>> 2
Enter elements>> 3
Enter elements of row number 2:
Enter elements>> 2
Enter elements>> 5
Enter elements>> 0
Addition of matrix:
  2   7   7
  2  12   3
subtraction of matrix:
  0   3   1
 -2   2   3
```

33.Find saddle points of a matrix:

Algorithm:

Name: Finding saddle point of a matrix

Input: i,j

Output: row, col,a[row][col]

Steps:

Step1: Start

Step2: Input i,j

Step3: Repeat Step4 for row=0 to i-1

Step4: Repeat Step5 for col=0 to j-1

Step5: Input a[row][col]

Step6: Repeat Step7 for row=0 to i-1

Step7: Repeat Step8,Step9,Step11,Step12 and Step14 for col=0 to j-1

Step8: min=a[row][0]

Step9: Repeat Step10 for k=1 to m-1

Step10: If a[row][k]<min then min=a[row][k]

Step11: max=a[0][col]

Step12: Repeat Step13 for k=1 to m-1

Step13: If a[row][k]>max then min=a[k][col]

Step14: If (a[row][col]==max && a[row][col]==min) then print row, col, a[row][col].

Step15: End

Program:

```
#include<stdio.h>
int main()
{
        int a[100][100],i,j,k,row,col,min,max;
        printf("Enter number of rows and columns in the matriix: ");
        scanf("%d%di",&i,&j);
        for(row=0;row<i;row++)
        {
           printf("Enter elements of row number: %d\n",row+1);
                for(col=0;col<j;col++)
```

```
                {
                   printf("Enter elements>> ");
                   scanf("%d",&a[row][col]);
        }
  }
  for(row=0;row<i;row++)
        {
                for(col=0;col<j;col++)
                {
                        min=a[row][0];
                        for(k=1;k<j;k++)
                        {
                                if(a[row][k]<min)
                                min=a[row][k];
                        }
                   max=a[0][col];
                        for(k=1;k<i;k++)
                        {
                                if(a[k][col]>max)
                                max=a[k][col];
                        }
                        if(a[row][col]==max&&a[row][col]==min)
                        {
                        printf("saddle point found at index number %d and %d",row,col);
                        printf("\nThe element is %d",a[row][col]);
                   }
    }
  }
```

Output:

Enter number of rows and columns in the matriix: 3 3

Enter elements of row number: 1

Enter elements>> 1

Enter elements>> 2

Enter elements>> 3

Enter elements of row number: 2

Enter elements>> 4

Enter elements>> 5

Enter elements>> 6

Enter elements of row number: 3

Enter elements>> 7

Enter elements>> 8

Enter elements>> 9

saddle point found at index number 2 and 0

The element is 7

34.Generate magic square:

Algorithm:

Name: Making a magic square

Input: n

Output: a[row][col]

Steps:

Step1: Start

Step2: Input n

Step3: Repeat Step4 for row=0 to n-1

Step4: Repeat Step5 for col=0 to n-1

Step5: a[row][col]=0

Step6: m=1

Step7: col=(n-1)/2

Step8: row=0

Step9: If m<=n*n then go to Step10, else go to Step20

Step10: If row=-1 && col!=n then row=n-1

Step11: Else

Step12: If row!=1 && col==n then col=0

Step13: Else

Step14: If (row=-1 && col=n) || (a[row][col]!=0)

Step15: row=row+2

Step16: col=col-1

Step17: End if

Step18: a[row][col]=m

Step19: m=m+1, row=row-1 and col=col+1 then go to Step9

Step20: Repeat Step21 for row=0 to n-1

Step21: Repeat Step22 for col=0 to n-1

Step22: Print a[row][col]

Step23: End

Program:

```
#include<stdio.h>
int main()
{
   int a[100][100];
   int row,col,count,n,m;
   printf("Enter the size of matrix: ");
   scanf("%d",&n);
   for(row=0; row<=n-1; row++)
   {
      for(col=0; col<=n-1; col++)
      {
         a[row][col]=0;
      }
   }
   m=1;
   col=(n-1)/2;
   row=0;
   while(m<=(n*n))
   {
      if((row==-1)&&(col!=n))
         row=n-1;
      else if((row!=-1)&&(col==n))
         col=0;
      else if((row==-1)&&(col==n)||(a[row][col]!=0))
      {
         row=row+2;
         col=col-1;
      }
      a[row][col]=m;
      m++;
      row--;
      col++;
   }
   printf("Magic square form>>\n");
   for(row=0; row<=n-1; row++)
   {
      for(col=0; col<=n-1; col++)
      {
         printf("%5d",a[row][col]);
      }
      printf("\n");
   }
}
```

Output:

Enter the size of matrix: 5

Magic square form>>

17 24 1 8 15

23 5 7 14 16

4 6 13 20 22

10 12 19 21 3

11 18 25 2 9

35.Multiplication of two matrices:

Algorithm:

Name: Multiplication of two matrix

Input: i,j,

Output:

Steps:

Step1: Start

Step2: Input I,j

Step3: Repeat Step4 for row=0 to i-1

Step4: Repeat Step5 for col=0 to j-1

Step5: Input a[row][col]

Step6: input k

Step7: Repeat Step8 for row=0 to i-1

Step8: Repeat Step9 for col=0 to k-1

Step9: Input c[row][col]

Step10: Repeat Step11 and Step16 for row=0 to i-1

Step11: Repeat Step13 and Step15 for row1=0 to k-1

Step12: s=0

Step13: Repeat Step14 for col=0 to j-1

Step14: s=s+a[row][col]*c[col][row1]

Step15: Print s

Step16: row=row+1

Step17: End

Program:

```
#include<stdio.h>
int main()
{
   int a[100][100],c[100][100];
   int row,col,i,j,s,row1,k;
//input of row&columm no taking>>
   printf("Enter total no of rows: ");
   scanf("%d",&i);
   printf("Enter total no of columms: ");
   scanf("%d",&j);
//input taking>>
   for(row=0; row<=i-1; row++)
   {
      printf("Enter row %d\n",row+1);
      for(col=0; col<=j-1; col++)
      {
         printf("enter elements-->");
         scanf("%d",&a[row][col]);
      }
   }
//input taking of 2nd matrix>>
   printf("enter no of columms of 2nd matrix>>\n");
   scanf("%d",&k);
   printf("Give elements of 2nd matrix-->\n");
   for(row=0; row<=j-1; row++)
   {
      printf("Enter row %d\n",row+1);
      for(col=0; col<=k-1; col++)
      {
         printf("enter elements-->");
         scanf("%d",&c[row][col]);
      }
   }
//multiplication >>
   printf("\multiplication of matrix-->\n");
   for(row=0; row<=i-1; row++)
   {
      for(row1=0; row1<=k-1; row1++)
      {
         s=0;
         for(col=0; col<=j-1; col++)
         {
            s=s+(a[row][col]*c[col][row1]);
```

```
        }
        printf("%3d",s);
      }
      row1++;
      printf("\n");
   }
}
```

Output:

```
Enter total no of rows: 2

Enter total no of columns: 3

Enter row 1

enter elements-->1

enter elements-->2

enter elements-->3

Enter row 2

enter elements-->4

enter elements-->5

enter elements-->6

enter no of columns of 2nd matrix>>

2

Give elements of 2nd matrix-->

Enter row 1

enter elements-->1

enter elements-->2

Enter row 2

enter elements-->3

enter elements-->4

Enter row 3

enter elements-->5

enter elements-->6

multiplication of matrix-->

 22 28

 49 64
```