//1. C PROGRAM FOR ARMSTRONG NUMBER

```
#include<stdio.h>
 int main()
{
  int n,r,sum=0,temp;
  printf("enter the number=");
  scanf("%d",&n);
  temp=n;
  while(n>0)
  {
   r=n%10;
   sum=sum+(r*r*r);
   n=n/10;
  }
  if(temp==sum)
   printf("armstrong  number ");
  else
   printf("not armstrong number");

}
```

OUTPUT:

```
enter the number=123
not armstrong number
```

//2. C PROGRAM FOR FIBONACCI SERIES USING RECURSION

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

```
}
```

OUTPUT:

```
Enter the number of terms: 5
Fibonacci Series: 0, 1, 1, 2, 3
```

//3. C PROGRAM FOR G.C.D OF TWO NUMBERS

```
#include <stdio.h>
int main()
{
   int n1, n2, i, gcd;
   printf("Enter two integers: ");
   scanf("%d %d", &n1, &n2);

   for(i=1; i <= n1 && i <= n2; ++i)
   {
      if(n1%i==0 && n2%i==0)
         gcd = i;
   }

   printf("G.C.D of %d and %d is %d", n1, n2, gcd);

   return 0;
}
```

OUTPUT:

```
Enter two integers: 2
8
G.C.D of 2 and 8 is 2
```

//4. C PROGRAM FOR LARGEST ELEMENT IN ARRAY

```
#include <stdio.h>
int main()
{
  int n;
  double arr[100];
  printf("Enter the number of elements : ");
  scanf("%d", &n);
  for (int i = 0; i < n; i++)
  {
    printf("Enter number%d: ", i + 1);
    scanf("%lf", &arr[i]);
  }
```

```
 for (int i = 1; i < n; i++)
 {
  if (arr[0] < arr[i])
  {
    arr[0] = arr[i];
  }
 }
 printf("Largest element = %.2lf", arr[0]);
 return 0;
}
```

OUTPUT:

```
Enter the number of elements : 4
Enter number1: 2
Enter number2: 3
Enter number3: 4
Enter number4: 2
Largest element = 4
```

//5. C PROGRAM FOR PRIME NUMBER

```
#include <stdio.h>
int main()
{
   int i,num, count = 0;
   printf("Enter the number: ");
   scanf("%d", &num);
   for(i = 1; i <= num; i++)
   {
     if(num % i == 0)
       count += 1;
   }
   if(count > 2)
      printf("%d is not prime", num);
   else
      printf("%d is prime", num);
return 0;
}
```

OUTPUT:

```
Enter the number: 5
5 is prime
```

//6. C PROGRAM FOR FACTORIAL

```
#include<stdio.h>
int main()
{
 int i,fact=1,number;
 printf("Enter a number: ");
  scanf("%d",&number);
    for(i=1;i<=number;i++){
      fact=fact*i;
  }
  printf("Factorial of %d is: %d",number,fact);
return 0;
}
```

OUTPUT:

```
Enter a number: 6
Factorial of 6 is: 720
```

//7. C PROGRAM FOR SELECTION SORT

```
#include <stdio.h>
int main()
{
   int arr[10];
   int i, j, position, swap,n;
   printf("Enter the number of elements: ");
   scanf("%d",&n);
   for (int i = 0; i < n; i++)
   {
    printf("Enter number%d: ", i + 1);
    scanf("%d", &arr[i]);
   }
   for (i = 0; i < (n - 1); i++)
   {
      position = i;
      for (j = i + 1; j < n; j++)
      {
         if (arr[position] > arr[j])
            position = j;
      }
      if (position != i)
      {
         swap = arr[i];
         arr[i] = arr[position];
         arr[position] = swap;
      }
   }
   for (i = 0; i < n; i++)
```

```
    printf("%d\t", arr[i]);
  return 0;
}
```

OUTPUT:

```
Enter the number of elements: 4
Enter number1: 3
Enter number2: 2
Enter number3: 1
Enter number4: 4
1       2       3       4
```

//8. C PROGRAM FOR BUBBLE SORT

```
#include <stdio.h>
int main()
{
 int array[100], n, c, d, swap;
 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]);
 for (c = 0 ; c < n - 1; c++)
 {
  for (d = 0 ; d < n - c - 1; d++)
  {
   if (array[d] > array[d+1])
   {
    swap       = array[d];
    array[d]   = array[d+1];
    array[d+1] = swap;
   }
  }
 }
 printf("Sorted list in ascending order:\n");
 for (c = 0; c < n; c++)
   printf("%d\n", array[c]);

 return 0;
}
```

OUTPUT:

```
Enter number of elements
5
Enter 5 integers
```

```
3
2
4
1
6
Sorted list in ascending order:
1
2
3
4
6
```

//9. C PROGRAM FOR PALINDROME

```
#include <stdio.h>
int main()
{
  int n, reversed = 0, remainder, original;
    printf("Enter an integer: ");
    scanf("%d", &n);
    original = n;
    while (n != 0)
    {
        remainder = n % 10;
        reversed = reversed * 10 + remainder;
        n /= 10;
    }
    if (original == reversed)
        printf("%d is a palindrome.", original);
    else
        printf("%d is not a palindrome.", original);

    return 0;
}
```

OUTPUT:

```
Enter an integer: 1234321
1234321 is a palindrome
```

//10. C PROGRAM FOR MATRIX MULTIPLICATION

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
  int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
  printf("enter the number of row=");
```

```
 scanf("%d",&r);
 printf("enter the number of column=");
 scanf("%d",&c);
 printf("enter the first matrix element=\n");
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
  {
   scanf("%d",&a[i][j]);
  }
 }
 printf("enter the second matrix element=\n");
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
  {
   scanf("%d",&b[i][j]);
  }
 }
 printf("multiply of the matrix=\n");
 for(i=0;i<r;i++)
 {
   for(j=0;j<c;j++)
   {
    mul[i][j]=0;
    for(k=0;k<c;k++)
    {
     mul[i][j]+=a[i][k]*b[k][j];
    }
   }
 }
 for(i=0;i<r;i++)
 {
  for(j=0;j<c;j++)
  {
    printf("%d\t",mul[i][j]);
  }
  printf("\n");
 }
 return 0;
}
```

OUTPUT:

```
enter the number of  row =2
enter the number of column =2
enter the first matrix element=
1
2
3
4
enter the second matrix element=
1
2
3
4
```

```
multiply of the matrix=
7       10
15      22
```

//11. C PRROGRAM FOR TO COPY ONE STRING TO THE ANOTHER

```
#include <stdio.h>
#include <string.h>
int main() {
   char str1[] = "Hello, world!";
   char str2[20];
   strcpy(str2, str1);
   printf("STRING 1: %s\n", str1);
   printf("STRING 2: %s\n", str2);

   return 0;
}
```

OUTPUT:

```
STRING 1: Hello, world!
STRING 2: Hello, world!
```

//12. C PROGRAM TO PERFORM BINARY SEARCH

```
#include <stdio.h>
int binarySearch(int arr[], int size, int element)
{
   int left = 0,mid;
   int right = size - 1;
   while (left <= right)
  {
      mid = left + (right - left) / 2;
      if (arr[mid] == element)
     {
         return mid;
      }
      else if (arr[mid] < element)
      {
         left = mid + 1;
      }
       else
       {
         right = mid - 1;
       }
   }
   return -1;
}
int main()
{
   int arr[] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
```

```
    int size = sizeof(arr) / sizeof(arr[0]);
    int element = 12;
    int result = binarySearch(arr, size,element);
    if (result != -1)
        {
       printf("Element %d found at index %d\n", element, result);
    }
        else
        {
       printf("Element %d not found in the array\n", element);
    }
    return 0;
}
```

OUTPUT:

```
Element 12 found at index 5
```

//13. C PROGRAM TO PRINT REVERSE OF A STRING

```
#include <stdio.h>
#include <string.h>
int main(void)
{
   char mystrg[60];
   int len, i;
   printf("insert the string to reverse: ");
   scanf( "%s", mystrg );
   len = strlen(mystrg);
   for(i = len - 1; i >= 0; i--)
         {
      printf("%c", mystrg[i]);
   }
   return 0;
}
```

OUTPUT:

```
insert the string to reverse: ABHINAY
YANIHBA
```

//14. C PROGRAM TO FIND THE LENGTH OF STRING

```
#include <stdio.h>
#include <string.h>
int main()
{
   char Str[1000];
   int i;
   printf("Enter the String: ");
   scanf("%s", Str);
   for (i = 0; Str[i] != '\0'; ++i);
   printf("Length of Str is %d", i);
```

```
   return 0;
}
```

OUTPUT:

```
Enter the String: ABHINAY
Length of Str is 7
```

//15. C PROGRAM TO PERFORM STRASSEN'S MATRIX MULTIPLICATION

```
 #include<stdio.h>
int main()
{
  int a[2][2], b[2][2], c[2][2], i, j;
  int m1, m2, m3, m4 , m5, m6, m7;

  printf("Enter the 4 elements of first matrix: ");
  for(i = 0;i < 2; i++)
     for(j = 0;j < 2; j++)
         scanf("%d", &a[i][j]);

  printf("Enter the 4 elements of second matrix: ");
  for(i = 0; i < 2; i++)
     for(j = 0;j < 2; j++)
         scanf("%d", &b[i][j]);

  printf("\nThe first matrix is\n");
  for(i = 0; i < 2; i++){
     printf("\n");
     for(j = 0; j < 2; j++)
         printf("%d\t", a[i][j]);
  }

  printf("\nThe second matrix is\n");
  for(i = 0;i < 2; i++){
     printf("\n");
     for(j = 0;j < 2; j++)
         printf("%d\t", b[i][j]);
  }

  m1= (a[0][0] + a[1][1]) * (b[0][0] + b[1][1]);
  m2= (a[1][0] + a[1][1]) * b[0][0];
  m3= a[0][0] * (b[0][1] - b[1][1]);
  m4= a[1][1] * (b[1][0] - b[0][0]);
  m5= (a[0][0] + a[0][1]) * b[1][1];
  m6= (a[1][0] - a[0][0]) * (b[0][0]+b[0][1]);
  m7= (a[0][1] - a[1][1]) * (b[1][0]+b[1][1]);

  c[0][0] = m1 + m4- m5 + m7;
  c[0][1] = m3 + m5;
  c[1][0] = m2 + m4;
  c[1][1] = m1 - m2 + m3 + m6;

   printf("\nAfter multiplication using Strassen's algorithm \n");
```

```
   for(i = 0; i < 2 ; i++){
      printf("\n");
      for(j = 0;j < 2; j++)
           printf("%d\t", c[i][j]);
   }

   return 0;
}
```

OUTPUT:

```
 Enter the 4 elements of first matrix: 1
2
3
4
Enter the 4 elements of second matrix: 3
2
1
3

The first matrix is

1       2
3       4
The second matrix is

3       2
1       3
After multiplication using Strassen's algorithm

5       8
13      18
```

//16. C PROGRAM TO PERFORM MERGE SORT

```
#include <stdio.h>
#include <stdlib.h>
 int main(void)
{
int i, n, j, k;
printf("Enter the size of the first array: ");
scanf("%d", &n);
int arr1[n];
printf("Enter the elements of the first array: \n");
for (i = 0; i < n; i++)
{
  scanf("%d", &arr1[i]);
    }
printf("Enter the size of the second array: ");
scanf("%d", &k);
int arr2[k];
printf("Enter the elements of the second array: \n");
for (j = 0; j < k; j++)
```

```
{
   scanf("%d", &arr2[j]);
   }
int arr3[n + k];
i = j = 0;
int in;
for (in = 0; in < n + k; in ++)
{
   if (i < n && j < k)
   {
   if (arr1[i] < arr2[j])
   {
   arr3[in] = arr1[i];
   i++;
   }
   else
   {
   arr3[in] = arr2[j];
   j++;
}
   }
   else if (i < n)
   {
   arr3[in] = arr1[i];
   i++;
   }
   else
   {
   arr3[in] = arr2[j];
   j++;
}
 }
   printf("The merged array is: \n");
   for (in = 0; in < n + k; in++)
   {
   printf("%d ", arr3[in]);
   }
   printf("\n");
   return 0;
}
```

OUTPUT:

```
Enter the size of the first array: 4
Enter the elements of the first array:
1
3
2
5
Enter the size of the second array: 4
```

Enter the elements of the second array:
5
2
4
3
The merged array is:
1 3 2 5 2 4 3 5

17.C PROGRAM FOR DIVIDE AND CONQUER STRATEGY

```
#include <stdio.h>
struct MinMax {
   int min;
   int max;
};
struct MinMax findMinMax(int arr[], int low, int high) {
   struct MinMax result, left, right, middle;
   if (low == high) {
      result.min = arr[low];
      result.max = arr[low];
      return result;
   }
   if (high - low == 1) {
      result.min = (arr[low] < arr[high]) ? arr[low] : arr[high];
      result.max = (arr[low] > arr[high]) ? arr[low] : arr[high];
      return result;
   }
   int mid = (low + high) / 2;
   left = findMinMax(arr, low, mid);
   right = findMinMax(arr, mid + 1, high);
   result.min = (left.min < right.min) ? left.min : right.min;
   result.max = (left.max > right.max) ? left.max : right.max;

   return result;
}

int main() {
   int n;
   printf("Enter the number of elements: ");
   scanf("%d", &n);

   int arr[n];
   printf("Enter the elements:\n");
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   struct MinMax result = findMinMax(arr, 0, n - 1);
   printf("Minimum: %d\n", result.min);
```

```
    printf("Maximum: %d\n", result.max);

    return 0;
}
```

OUTPUT:

Enter the number of elements: 6
Enter the elements:
2
4
3
6
3
2
Minimum: 2
Maximum: 6

//18.C PROGRAM TO GENERATE ALL PRIME NUMBERS

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

void sieveOfEratosthenes(int n) {
    bool isPrime[n + 1];
    for (int i = 0; i <= n; i++) {
        isPrime[i] = true;
    }
    for (int p = 2; p * p <= n; p++) {
        if (isPrime[p]) {
            for (int i = p * p; i <= n; i += p) {
                isPrime[i] = false;
            }
        }
    }
    printf("Prime numbers in the range 2 to %d:\n", n);
    for (int i = 2; i <= n; i++) {
        if (isPrime[i]) {
            printf("%d ", i);
        }
    }
    printf("\n");
}

int main() {
    int range;
    printf("Enter the range to generate prime numbers: ");
    scanf("%d", &range);
```

```
    if (range < 2) {
        printf("There are no prime numbers in the given range.\n");
    } else {
        sieveOfEratosthenes(range);
    }

    return 0;
}
```

OUTPUT:

```
Enter the range to generate prime numbers: 3
Prime numbers in the range 2 to 3:
2 3
```

//19.C PROGRAM TO PERFORM KNAPSACK PROBLEM USING GREEDY TECHNIQUES

```
include<stdio.h>
void knapsack(int n, float weight[], float profit[], float capacity)
 {
   float x[20], tp = 0;
   int i, j, u;
   u = capacity;
   for (i = 0; i < n; i++)
    x[i] = 0.0;

   for (i = 0; i < n; i++)
   {
    if (weight[i] > u)
          break;
    else
          {
    x[i] = 1.0;
    tp = tp + profit[i];
    u = u - weight[i];
    }
}

   if (i < n)
   x[i] = u / weight[i];
   tp = tp + (x[i] * profit[i]);
   printf("\nMaximum profit is:- %f", tp);

}
 int main()
{
   float weight[20], profit[20], capacity;
```

```
   int num, i, j;
   float ratio[20], temp;

   printf("\nEnter the no. of items:- ");
   scanf("%d", &num);

   printf("\nEnter the wts and profits of each item:- ");
   for (i = 0; i < num; i++)
{
    scanf("%f %f", &weight[i], &profit[i]);
   }
printf("\nEnter the capacity of knapsack:- ");
scanf("%f", &capacity);
for (i = 0; i < num; i++)
{
    ratio[i] = profit[i] / weight[i];
   }
 for (i = 0; i < num; i++)
{
for (j = i + 1; j < num; j++)
{
    if (ratio[i] < ratio[j])
          {
    temp = ratio[j];
    ratio[j] = ratio[i];
    ratio[i] = temp;

    temp = weight[j];
    weight[j] = weight[i];
    weight[i] = temp;

    temp = profit[j];
    profit[j] = profit[i];
    profit[i] = temp;
}
 }
  }
knapsack(num, weight, profit, capacity);
return(0);
}
```

OUTPUT:

```
Enter the no. of items:- 5

Enter the wts and profits of each item:- 10
20
30
40
50
2
```

3
4
5
6

Enter the capacity of knapsack:- 100

Maximum profit is:- 72.000000

//20.C PROGRAM TO PERFORM MST USING GREEDY TECHNIQUES

```
#include<stdio.h>
#include<conio.h>
int n, cost[10][10];
void prim() {
  int i, j, startVertex, endVertex;
  int k, nr[10], temp, minimumCost = 0, tree[10][3];
  temp = cost[0][0];
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      if (temp > cost[i][j]) {
        temp = cost[i][j];
        startVertex = i;
        endVertex = j;
      }
    }
  }
  tree[0][0] = startVertex;
  tree[0][1] = endVertex;
  tree[0][2] = temp;
  minimumCost = temp;
  for (i = 0; i < n; i++) {
    if (cost[i][startVertex] < cost[i][endVertex])
      nr[i] = startVertex;
    else
      nr[i] = endVertex;
  }
  nr[startVertex] = 100;
  nr[endVertex] = 100;
  temp = 99;
  for (i = 1; i < n - 1; i++) {
    for (j = 0; j < n; j++) {
      if (nr[j] != 100 && cost[j][nr[j]] < temp) {
        temp = cost[j][nr[j]];
        k = j;
      }
    }
    tree[i][0] = k;
```

```
    tree[i][1] = nr[k];
    tree[i][2] = cost[k][nr[k]];
    minimumCost = minimumCost + cost[k][nr[k]];
    nr[k] = 100;
    for (j = 0; j < n; j++) {
      if (nr[j] != 100 && cost[j][nr[j]] > cost[j][k])
        nr[j] = k;
    }
    temp = 99;
  }
  printf("\nThe min spanning tree is: ");
  for (i = 0; i < n - 1; i++) {
    for (j = 0; j < 3; j++)
      printf("%d", tree[i][j]);
    printf("\n");
  }

  printf("\nMin cost : %d", minimumCost);
}

int main() {
  int i, j;
  printf("\nEnter the no. of vertices :");
  scanf("%d", &n);
  printf("\nEnter the costs of edges in matrix form :");
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++) {
      scanf("%d", &cost[i][j]);
    }
  printf("\nThe matrix is :\n ");
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      printf("%d\t", cost[i][j]);
    }
    printf("\n");
  }
  prim();
  getch();
  return 0;
}
```

OUTPUT:

```
Enter the no. of vertices :4

Enter the costs of edges in matrix form :2
3
4
5
6
2
```

4
2
45
5
6
3
2
4
5
6

The matrix is :

2	**3**	**4**	**5**
6	**2**	**4**	**2**
45	**5**	**6**	**3**
2	**4**	**5**	**6**

//21.C PROGRAM FOR OPTIMAL BINARY SEARCH TREE

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

#define MAX_KEYS 10
int optimalBST(int keys[], int freq[], int n) {
   int cost[n + 1][n + 1];
   for (int i = 0; i < n; i++) {
      cost[i][i] = freq[i];
   }
   for (int chainLength = 2; chainLength <= n; chainLength++) {
      for (int i = 0; i <= n - chainLength + 1; i++) {
         int j = i + chainLength - 1;
         cost[i][j] = INT_MAX;
         for (int r = i; r <= j; r++) {
            int left = (r > i) ? cost[i][r - 1] : 0;
            int right = (r < j) ? cost[r + 1][j] : 0;
            int sumFreq = 0;
            for (int k = i; k <= j; k++) {
               sumFreq += freq[k];
            }
            int currentCost = left + right + sumFreq;
            if (currentCost < cost[i][j]) {
               cost[i][j] = currentCost;
            }
         }
      }
   }

   return cost[0][n - 1];
```

```
}

int main() {
   int n;
   printf("Enter the number of keys: ");
   scanf("%d", &n);

   int keys[MAX_KEYS];
   int freq[MAX_KEYS];

   printf("Enter the keys and their corresponding frequencies:\n");
   for (int i = 0; i < n; i++) {
      scanf("%d %d", &keys[i], &freq[i]);
   }

   int result = optimalBST(keys, freq, n);
   printf("Optimal cost of binary search tree: %d\n", result);

   return 0;
}
```

OUTPUT:

```
Enter the number of keys: 4
Enter the keys and their corresponding frequencies:
10
20
30
40
1
2
3
4
Optimal cost of binary search tree: 94
```

//22.C PROGRAM TO FIND BINOMIAL COEFFICIENT OF A GIVEN NUMBER

```
#include <stdio.h>
int biCo(int n, int k)
{
        if (k > n)
                return 0;
        if (k == 0 || k == n)
                return 1;
        return biCo(n - 1, k - 1)
                + biCo(n - 1, k);
}
int main()
{
```

```
        int n , k;
        printf (" enter the n value:");
        scanf("%d",&n);
        printf (" enter the k value:");
        scanf("%d",&k);
        printf("Value of C(%d, %d) is %d ", n, k,biCo(n, k));
        return 0;
}
```

OUTPUT:

```
enter the n value:5
 enter the k value:3
Value of C(5, 3) is 10
```

23.C PROGRAM TO REVERSE A NUMBER

```
#include<stdio.h>
int main()
{
 int n,rev=0,rem;
 printf("enter the number");
 scanf("%d",&n);
 while(n!=0)
 {
  rem=n%10;
  rev=rev*10+rem;
  n/=10;
 }
 printf("The reversed number is %d",rev);
 return 0;
}
```

OUTPUT:

```
Enter the number1234
The reversed number is 4321
```

24.C PROGRAM FOR PERFECT NUMBER

```
#include<stdio.h>
int main()
{
int num, rem, sum = 0, i;
printf("Enter a number\n");
scanf("%d", &num);
for(i = 1; i < num; i++)
{
  rem = num % i;
  if (rem == 0)
  {
```

```
   sum = sum + i;
  }
}
if (sum == num)
   printf("It is a Perfect Number");
else
   printf("It is not a Perfect Number");
return 0;
}
```

OUTPUT:

```
Enter a number
10
It is not a Perfect Number
```

25.C PROGRAM TO PRINT THE FOLLOWING PATTERN

1
12
123
1234

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

OUTPUT:

```
Enter number of rows: 4
1
12
123
1234
```

26.C PROGRAM FOR SUM OF DIGITS

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

```
int main()
{
 int n,sum=0,digit;
 printf("Enter a number:");
 scanf("%d",&n);
 while(n>0)
 {
  digit=n%10;
  sum=sum+digit;
  n=n/10;
 }
 printf("Sum is %d",sum);
 return 0;
}
```

OUTPUT:

```
Enter a number:453
Sum is 12
```

27.C PROGRAM TO PRINT PASCAL'S TRIANGLE

```
#include <stdio.h>
void main()
{
   int no_row,c=1,blk,i,j;
   printf("Input number of rows: ");
   scanf("%d",&no_row);
   for(i=0;i<no_row;i++)
   {
      for(blk=1;blk<=no_row-i;blk++)
      printf("  ");
      for(j=0;j<=i;j++)
      {
         if (j==0||i==0)
            c=1;
         else
            c=c*(i-j+1)/j;
         printf("% 4d",c);
      }
      printf("\n");
   }
}
```

OUTPUT:

```
Input number of rows: 4
          1
        1   1
      1   2   1
    1   3   3   1
```

28.C PROGRAM FOR FLOYD'S ALGORITHM

```
#include<stdio.h>
int min(int,int);
void floyds(int p[10][10],int n)
{
        int i,j,k;
        for (k=1;k<=n;k++)
          for (i=1;i<=n;i++)
           for (j=1;j<=n;j++)
            if(i==j)
             p[i][j]=0; else
             p[i][j]=min(p[i][j],p[i][k]+p[k][j]);
}
int min(int a,int b)
{
        if(a<b)
          return(a); else
          return(b);
}
void main()
{
        int p[10][10],w,n,e,u,v,i,j;
        printf("\n Enter the number of vertices:");
        scanf("%d",&n);
        printf("\n Enter the number of edges:\n");
        scanf("%d",&e);
        for (i=1;i<=n;i++)
      {
                for (j=1;j<=n;j++)
                  p[i][j]=999;
        }
        for (i=1;i<=e;i++)
      {
                printf("\n Enter the end vertices of edge%d with its weight \n",i);
                scanf("%d%d%d",&u,&v,&w);
                p[u][v]=w;
        }
        printf("\n Matrix of input data:\n");
        for (i=1;i<=n;i++)
      {
                for (j=1;j<=n;j++)
                  printf("%d \t",p[i][j]);
                printf("\n");
        }
        floyds(p,n);
        printf("\n Transitive closure:\n");
        for (i=1;i<=n;i++)
      {
                for (j=1;j<=n;j++)
```

```
                printf("%d \t",p[i][j]);
            printf("\n");
        }
        printf("\n The shortest paths are:\n");
        for (i=1;i<=n;i++)
          for (j=1;j<=n;j++)
    {
            if(i!=j)
               printf("\n <%d,%d>=%d",i,j,p[i][j]);
        }
}
```

OUTPUT:

```
 Enter the number of vertices:4

 Enter the number of edges:
4

 Enter the end vertices of edge1 with its weight
1
2
10

 Enter the end vertices of edge2 with its weight
2
3
15

 Enter the end vertices of edge3 with its weight
3
4
10

 Enter the end vertices of edge4 with its weight
4
1
15

 Matrix of input data:
999     10      999     999
999     999     15      999
999     999     999     10
15      999     999     999

 Transitive closure:
0       10      25      35
40      0       15      25
25      35      0       10
15      25      40      0
```

The shortest paths are:

<1,2>=10
<1,3>=25
<1,4>=35
<2,1>=40
<2,3>=15
<2,4>=25
<3,1>=25
<3,2>=35
<3,4>=10
<4,1>=15
<4,2>=25
<4,3>=40

29.C PROGRAM TO PERFORM N QUEENS PROBLEM USING BACK TRACKING

```
#include<stdio.h>
#include<math.h>
int board[20],count;
int main()
{
  int n,i,j;
  void queen(int row,int n);
  printf("\n\nEnter number of Queens:");
  scanf("%d",&n);
  queen(1,n);
   return 0;
}
void print(int n)
{
   int i,j;
   printf("\n\nSolution %d:\n\n",++count);
   for(i=1;i<=n;++i)
     printf("\t%d",i);
  for(i=1;i<=n;++i)
  {
    printf("\n\n%d",i);
      for(j=1;j<=n;++j)
      {
       if(board[i]==j)
         printf("\tQ");
       else
         printf("\t-");
      }
  }
 }
int place(int row,int column)
```

```
{
  int i;
  for(i=1;i<=row-1;++i)
  {
    if(board[i]==column)
      return 0;
    else
      if(abs(board[i]-column)==abs(i-row))
       return 0;
  }
  return 1;
}
void queen(int row,int n)
{

   int column;
   for(column=1;column<=n;++column)
   {
     if(place(row,column))
     {
       board[row]=column;
       if(row==n)
          print(n);
       else
          queen(row+1,n);
     }
   }
}
```

OUTPUT:

Enter number of Queens:4
Solution 1:

	1	2	3	4
1	-	Q	-	-
2	-	-	-	Q
3	Q	-	-	-
4	-	-	Q	-

Solution 2:

	1	2	3	4
1	-	-	Q	-
2	Q	-	-	-

3 - - - Q

4 - Q - -

30.C PROGRAM TO PRINT MINIMUM AND MAXIMUM SEQUENCE FOR ALL NUMBERS IN LIST

```
#include <stdio.h>
void minimumSort(int arr[], int n)
{
   for (int i = 0; i < n - 1; i++)
         {
      for (int j = 0; j < n - i - 1; j++)
                {
         if (arr[j] > arr[j + 1])
                        {
            int temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
         }
      }
   }
}
void maximumSort(int arr[], int n)
{
   for (int i = 0; i < n - 1; i++)
         {
      for (int j = 0; j < n - i - 1; j++)
                {
         if (arr[j] < arr[j + 1])
                        {
            int temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
         }
      }
   }
}
int main()
{
   int n;
   printf("Enter the number of elements: ");
   scanf("%d", &n);
   int arr[n];
   printf("Enter %d elements:\n", n);
   for (int i = 0; i < n; i++)
         {
      scanf("%d", &arr[i]);
```

```
    }
    minimumSort(arr, n);
    printf("Minimum Sequence:\n");
    for (int i = 0; i < n; i++)
            {
        printf("%d ", arr[i]);
    }
    printf("\n");
    maximumSort(arr, n);
    printf("Maximum Sequence:\n");
    for (int i = 0; i < n; i++)
            {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

OUTPUT:

```
Enter the number of elements: 5
Enter 5 elements:
1
2
3
4
5
Minimum Sequence:
1 2 3 4 5
Maximum Sequence:
5 4 3 2 1
```

31.C PROGRAM FOR TRAVELLING SALESMAN PROBLEM

```
#include<stdio.h>
int ary[10][10],completed[10],n,cost=0;
void takeInput()
{
int i,j;
printf("Enter the number of villages: ");
scanf("%d",&n);
printf("\nEnter the Cost Matrix\n");
for(i=0;i < n;i++)
{
printf("\nEnter Elements of Row: %d\n",i+1);
for( j=0;j < n;j++)
scanf("%d",&ary[i][j]);
completed[i]=0;
}
printf("\n\nThe cost list is:");
for( i=0;i < n;i++)
{
```

```
printf("\n");
for(j=0;j < n;j++)
printf("\t%d",ary[i][j]);
}
}
void mincost(int city)
{
int i,ncity;
completed[city]=1;
printf("%d--->",city+1);
ncity=least(city);
if(ncity==999)
{
ncity=0;
printf("%d",ncity+1);
cost+=ary[city][ncity];
return;
}
mincost(ncity);
}
int least(int c)
{
int i,nc=999;
int min=999,kmin;
for(i=0;i < n;i++)
{
if((ary[c][i]!=0)&&(completed[i]==0))
if(ary[c][i]+ary[i][c] < min)
{
min=ary[i][0]+ary[c][i];
kmin=ary[c][i];
nc=i;
}
}
if(min!=999)
cost+=kmin;
return nc;
}
int main()
{
takeInput();
printf("\n\nThe Path is:\n");
mincost(0);
printf("\n\nMinimum cost is %d\n ",cost);
return 0;
}
```

OUTPUT:

```
Enter the number of villages: 3
Enter the Cost Matrix
```

```
Enter Elements of Row: 1
1
2
31

2

Enter Elements of Row: 2

1
2
3
41
1

2

Enter Elements of Row: 3
1

2

3


The cost list is:
        1       1       2
        1       1       2
        1       2       3

The Path is:
1--->2--->3--->1

Minimum cost is 4
```

32.C PROGRAM TO FIND OPTIMAL COST USING APPROPRIATE ALGORITHM

```
#include <stdio.h>
#include <limits.h>
#define MAX_KEYS 10
int optimalCostBST(int keys[], int freq[], int n) {
   int cost[n][n];
   for (int i = 0; i < n; i++) {
      cost[i][i] = freq[i];
   }
   for (int chain_len = 2; chain_len <= n; chain_len++) {
      for (int start = 0; start < n - chain_len + 1; start++) {
         int end = start + chain_len - 1;
         cost[start][end] = INT_MAX;
         for (int root = start; root <= end; root++) {
            int left_cost = (root > start) ? cost[start][root - 1] : 0;
```

```
            int right_cost = (root < end) ? cost[root + 1][end] : 0;
            int current_cost = left_cost + right_cost + freq[root];
            if (current_cost < cost[start][end]) {
                cost[start][end] = current_cost;
            }
        }
    }
  }

  return cost[0][n - 1];
}
int main() {
  int keys[MAX_KEYS], freq[MAX_KEYS];
  int n;
  printf("Enter the number of keys (maximum %d): ", MAX_KEYS);
  scanf("%d", &n);
  printf("Enter the keys in sorted order:\n");
  for (int i = 0; i < n; i++) {
    scanf("%d", &keys[i]);
  }
  printf("Enter the frequencies of the keys:\n");
  for (int i = 0; i < n; i++) {
    scanf("%d", &freq[i]);
  }
  int optimal_cost = optimalCostBST(keys, freq, n);
  printf("Optimal cost of the Binary Search Tree is: %d\n", optimal_cost);
  return 0;
}
```

OUTPUT:

```
Enter the number of keys (maximum 10): 3
Enter the keys in sorted order:
1
2
3
Enter the frequencies of the keys:
3
2
4
Optimal cost of the Binary Search Tree is: 9
```

33.WRITE A PROGRAM TO INSERT NUMBER INTO A LIST

```
#include <stdio.h>
int main()
{
  int arr[100] = { 0 };
  int i, x, pos, n;
  printf("Enter number of elements: ");
  scanf("%d",&n);
  printf("Enter the elements: ");
```

```
    for (i = 0; i < n; i++)
    {
      scanf("%d",&arr[i]);
    }
    for (i = 0; i < n; i++)
    {
       printf("%d ", arr[i]);
    }
    printf("\n");
    printf("Enter the element to insert: ");
    scanf("%d",&x);
    printf("Enter the position: ");
    scanf("%d",&pos);
    n++;
    for (i = n - 1; i >= pos; i--)
       arr[i] = arr[i - 1];
    arr[pos - 1] = x;
    for (i = 0; i < n; i++)
       printf("%d ", arr[i]);
    printf("\n");
    if(pos>n)
      printf("Entered postion is not valid");
    return 0;
}
```

OUTPUT:

```
Enter number of elements: 7
Enter the elements: 1
2
3
4
5
6
7
1 2 3 4 5 6 7
Enter the element to insert: 37
Enter the position: 7
1 2 3 4 5 6 37 7
```

34.WRITE A PROGRAM TO PRINT ALL THE FACTORS OF A NUMBER

```
#include <stdio.h>
int main()
{
    int num, i;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
```

```
    printf("Factors of %d are: ", num);
    for (i = 1; i <= num; ++i)
    {
        if (num % i == 0)
        {
            printf("%d ", i);
        }
    }
    return 0;
}
```

OUTPUT:

```
Enter a positive integer: 10
Factors of 10 are: 1 2 5 10
```

35.WRITE A PROGRAM TO PERFORM LINEAR SEARCH

```
#include<stdio.h>
int main()
{
 int a[20],i,x,n;
 printf("Enter number of elements: ");
 scanf("%d",&n);
 printf("Enter the elements: ");
 for(i=0;i<n;i++)
 {
  scanf("%d",&a[i]);
 }
 printf("enter the key element: ");
 scanf("%d",&x);
 for(i=0;i<n;i++)
 {
   if(a[i]==x)
   break;
 }
 if(i<n)
  printf("Element found in position %d",i);
 else
  printf("Element not found");
 return 0;
}
```

OUTPUT:

```
Enter number of elements: 5
Enter the elements: 1
3
4
2
```

```
4
enter the key element: 2
Element found in position 3
```

36.WRITE A PROGRAM TO COMPUTE CONTAINER LOADING PROBLEM

```
#include <stdio.h>
#define MAX_CONTAINERS 100
#define MAX_ITEMS 100
struct Container
{
   int capacity;
   int currentLoad;
};
struct Item
{
   int weight;
   int containerIndex;
};
void initializeContainers(struct Container containers[], int numContainers, int
containerCapacity)
{
   for (int i = 0; i < numContainers; i++)
         {
      containers[i].capacity = containerCapacity;
      containers[i].currentLoad = 0;
   }
}
void loadItems(struct Container containers[], struct Item items[], int numItems)
{
   for (int i = 0; i < numItems; i++)
         {
      for (int j = 0; j < MAX_CONTAINERS; j++)
               {
         if (containers[j].currentLoad + items[i].weight <= containers[j].capacity)
                  {
            containers[j].currentLoad += items[i].weight;
            items[i].containerIndex = j;
            break;
         }
      }
   }
}

void printContainerContents(struct Container containers[], int numContainers)
{
   for (int i = 0; i < numContainers; i++)
         {
```

```
        printf("Container %d: Load %d / Capacity %d\n", i, containers[i].currentLoad,
containers[i].capacity);
    }
}

int main()
{
    struct Container containers[MAX_CONTAINERS];
    struct Item items[MAX_ITEMS];
    int numContainers, containerCapacity, numItems;
    printf("Enter the number of containers: ");
    scanf("%d", &numContainers);
    printf("Enter the capacity of each container: ");
    scanf("%d", &containerCapacity);
    printf("Enter the number of items: ");
    scanf("%d", &numItems);
    printf("Enter the weight of each item:\n");
    for (int i = 0; i < numItems; i++)
            {
        scanf("%d", &items[i].weight);
        items[i].containerIndex = -1;
    }
    initializeContainers(containers, numContainers, containerCapacity);
    loadItems(containers, items, numItems);
    printf("\nContainer Loading Result:\n");
    printContainerContents(containers, numContainers);
    return 0;
}
```

OUTPUT:

```
Enter the number of containers: 3
Enter the capacity of each container: 20
Enter the number of items: 4
Enter the weight of each item:
10
15
20
12

Container Loading Result:
Container 0: Load 10 / Capacity 20
Container 1: Load 15 / Capacity 20
Container 2: Load 20 / Capacity 20
```

37.WRITE A PROGRAM TO FIND OUT HAMILTONIAN CIRCUIT USING BACK TRACKING

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

```
#define MAX_VERTICES 10
bool isSafe(int v, int path[], int graph[MAX_VERTICES][MAX_VERTICES], int pathLength,
int pos) {
   if (!graph[path[pos - 1]][v]) {
      return false;
   }

   for (int i = 0; i < pathLength; i++) {
      if (path[i] == v) {
         return false;
      }
   }

   return true;
}

bool hamiltonianCycleUtil(int graph[MAX_VERTICES][MAX_VERTICES], int path[], int
pathLength, int totalVertices) {
   if (pathLength == totalVertices) {
      if (graph[path[pathLength - 1]][path[0]]) {
         return true;
      }
      return false;
   }

   for (int v = 1; v < totalVertices; v++) {
      if (isSafe(v, path, graph, pathLength, pathLength)) {
         path[pathLength] = v;

         if (hamiltonianCycleUtil(graph, path, pathLength + 1, totalVertices)) {
            return true;
         }

         path[pathLength] = -1;
      }
   }

   return false;
}

bool hamiltonianCycle(int graph[MAX_VERTICES][MAX_VERTICES], int totalVertices) {
   int path[MAX_VERTICES];
   for (int i = 0; i < totalVertices; i++) {
      path[i] = -1;
   }
   path[0] = 0;

   if (!hamiltonianCycleUtil(graph, path, 1, totalVertices)) {
      printf("No Hamiltonian cycle exists.\n");
      return false;
   }

   printf("Hamiltonian cycle exists:\n");
   for (int i = 0; i < totalVertices; i++) {
```

```
      printf("%d ", path[i]);
   }
   printf("%d\n", path[0]);
   return true;
}

int main() {
   int totalVertices;

   printf("Enter the number of vertices in the graph: ");
   scanf("%d", &totalVertices);

   int graph[MAX_VERTICES][MAX_VERTICES];
   printf("Enter the adjacency matrix for the graph:\n");
   for (int i = 0; i < totalVertices; i++) {
      for (int j = 0; j < totalVertices; j++) {
         scanf("%d", &graph[i][j]);
      }
   }

   hamiltonianCycle(graph, totalVertices);

   return 0;
}
```

OUTPUT:

```
Enter the number of vertices in the graph: 3
Enter the adjacency matrix for the graph:
1
3
2
4
5
2
4
4
5
Hamiltonian cycle exists:
0 1 2 0
```

38.WRITE A PROGRAM TO PERFORM GRAPH COLOURING PROBLEM USING BACK TRACKING

```
#include <stdio.h>
#include <stdbool.h>
#define MAX_VERTICES 20
void printSolution(int colors[], int num_vertices);
bool isSafe(int vertex, int graph[][MAX_VERTICES], int colors[], int num_vertices, int
color)
 {
   for (int i = 0; i < num_vertices; i++)
   {
      if (graph[vertex][i] && colors[i] == color)
```

```
      {
          return false;
       }
   }
   return true;
}
bool graphColoringUtil(int graph[][MAX_VERTICES], int num_vertices, int m, int colors[],
int vertex)
{
   if (vertex == num_vertices)
    {
      return true;
     }
   for (int color = 1; color <= m; color++)
   {
      if (isSafe(vertex, graph, colors, num_vertices, color))
      {
          colors[vertex] = color;
          if (graphColoringUtil(graph, num_vertices, m, colors, vertex + 1))
         {
             return true;
          }
          colors[vertex] = 0;
       }
     }

   return false;
}
bool graphColoring(int graph[][MAX_VERTICES], int num_vertices, int m)
{
   int colors[MAX_VERTICES];
   for (int i = 0; i < num_vertices; i++)
  {
      colors[i] = 0;
   }
   if (!graphColoringUtil(graph, num_vertices, m, colors, 0))
  {
      return false;
   }
   printf("Graph can be colored using %d colors.\n", m);
   printf("Coloring of vertices:\n");
   printSolution(colors, num_vertices);
   return true;
}
void printSolution(int colors[], int num_vertices)
 {
   for (int i = 0; i < num_vertices; i++) {
      printf("Vertex %d: Color %d\n", i, colors[i]);
   }
}
```

```
int main()
{
   int num_vertices, m;
   printf("Enter the number of vertices (max %d): ", MAX_VERTICES);
   scanf("%d", &num_vertices);
   int graph[MAX_VERTICES][MAX_VERTICES];
   printf("Enter the adjacency matrix for the graph:\n");
   for (int i = 0; i < num_vertices; i++)
   {
      for (int j = 0; j < num_vertices; j++)
      {
         scanf("%d", &graph[i][j]);
      }
    }
   printf("Enter the number of colors: ");
   scanf("%d", &m);
   if (m < 1)
  {
      printf("Number of colors should be at least 1.\n");
      return 1;
   }
   if (!graphColoring(graph, num_vertices, m))
  {
      printf("Graph cannot be colored with the given constraints.\n");
   }
   return 0;
}
```

OUTPUT:

```
Enter the number of vertices (max 20): 4
Enter the adjacency matrix for the graph:
0 2 3 4
1 2 3 4
4 3 2 1
0 3 4 2
Enter the number of colors: 4
Graph can be colored using 4 colors.
Coloring of vertices:
Vertex 0: Color 1
Vertex 1: Color 2
Vertex 2: Color 3
Vertex 3: Color 1
```

---

39. WRITE A PROGRAM TO PERFORM SUM OF SUBSETS PROBLEM USING BACK TRACKING

```
#include <stdio.h>
#include <stdbool.h>
void generateSubsets(int arr[], int n, bool subset[], int index, int targetSum, int currentSum) {
   if (index == n) {
```

```
        if (currentSum == targetSum) {
            printf("Subset with target sum %d: {", targetSum);
            for (int i = 0; i < n; i++) {
                if (subset[i]) {
                    printf(" %d", arr[i]);
                }
            }
            printf(" }\n");
        }
        return;
    }
    subset[index] = true;
    generateSubsets(arr, n, subset, index + 1, targetSum, currentSum + arr[index]);
    subset[index] = false;
    generateSubsets(arr, n, subset, index + 1, targetSum, currentSum);
}
int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the elements:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int targetSum;
    printf("Enter the target sum: ");
    scanf("%d", &targetSum);
    bool subset[n];
    generateSubsets(arr, n, subset, 0, targetSum, 0);
    return 0;
}
```

OUTPUT:

```
Enter the number of elements: 4
Enter the elements:
1
2
3
4
Enter the target sum: 7
Subset with target sum 7: { 1 2 4 }
Subset with target sum 7: { 3 4 }
```

40.WRITE A PROGRAM TO PERFORM ASSIGNMENT PROBLEM USING BRANCH AND BOUND

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

```
#define MAX_SIZE 100
#define INF 9999
int numWorkers;
int numJobs;
int costMatrix[MAX_SIZE][MAX_SIZE];
bool assigned[MAX_SIZE];
int minCost = INF;
int finalAssignment[MAX_SIZE];
void printAssignment() {
   printf("Optimal Assignment:\n");
   for (int i = 0; i < numWorkers; i++) {
      printf("Worker %d -> Job %d\n", i + 1, finalAssignment[i] + 1);
   }
}
void updateMinCost(int assignment[MAX_SIZE]) {
   int totalCost = 0;
   for (int i = 0; i < numWorkers; i++) {
      totalCost += costMatrix[i][assignment[i]];
   }

   if (totalCost < minCost) {
      minCost = totalCost;
      for (int i = 0; i < numWorkers; i++) {
         finalAssignment[i] = assignment[i];
      }
   }
}
void branchAndBound(int worker, int currentCost, int assignment[MAX_SIZE]) {
   if (worker == numWorkers) {
      updateMinCost(assignment);
      return;
   }
   for (int job = 0; job < numJobs; job++) {
      if (!assigned[job]) {
         int newCost = currentCost + costMatrix[worker][job];

         if (newCost < minCost) {
            assigned[job] = true;
            assignment[worker] = job;

            branchAndBound(worker + 1, newCost, assignment);

            assigned[job] = false;
         }
      }
   }
}
int main() {
   printf("Enter the number of workers: ");
   scanf("%d", &numWorkers);
   printf("Enter the number of jobs: ");
   scanf("%d", &numJobs);
   printf("Enter the cost matrix (%d x %d):\n", numWorkers, numJobs);
   for (int i = 0; i < numWorkers; i++) {
      for (int j = 0; j < numJobs; j++) {
```

```
            scanf("%d", &costMatrix[i][j]);
        }
    }
    int assignment[MAX_SIZE];
    for (int i = 0; i < numWorkers; i++) {
        assigned[i] = false;
    }
    branchAndBound(0, 0, assignment);
    printf("Minimum Cost: %d\n", minCost);
    printAssignment();
    return 0;
}
```

OUTPUT:

```
Enter the number of workers: 4
Enter the number of jobs: 4
Enter the cost matrix (4 x 4):
1 2 3 4
0 1 2 3
0 2 1 2
0 1 3 2
Minimum Cost: 5
Optimal Assignment:
Worker 1 -> Job 1
Worker 2 -> Job 2
Worker 3 -> Job 3
Worker 4 -> Job 4
```