1.**TRAVELLING SALES MAN**

```
#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); //passing 0 because starting vertex
printf("\n\nMinimum cost is %d\n ",cost);
return 0;
}
```

2.SUM OF SUBSETS

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

#define MAX_SIZE 10

void print_subset(int subset[], int size) {
   printf("{ ");
   for (int i = 0; i < size; i++) {
      printf("%d ", subset[i]);
   }
   printf("}\n");
}

void backtrack(int set[], int subset[], int set_size, int subset_size, int sum, int target_sum, int
index) {
   if (sum == target_sum) {
      print_subset(subset, subset_size);
      return;
   }
   if (sum > target_sum || index >= set_size) {
      return;
   }
   // include the current element
   subset[subset_size] = set[index];
```

```
    backtrack(set, subset, set_size, subset_size + 1, sum + set[index], target_sum, index + 1);
    // exclude the current element
    backtrack(set, subset, set_size, subset_size, sum, target_sum, index + 1);
}

int main() {
    int set[MAX_SIZE];
    int set_size, target_sum;

    printf("Enter the size of the set (<= %d): ", MAX_SIZE);
    scanf("%d", &set_size);

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

    printf("Enter the target sum: ");
    scanf("%d", &target_sum);

    int subset[MAX_SIZE];
    backtrack(set, subset, set_size, 0, 0, target_sum, 0);

    return 0;
}
```

3.SUM OF DIGITS

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

4.**STRING COPIED**

```
#include<stdio.h>
#include<string.h>
int main()
```

```
{
	char s[20],a[20];
	int l,i;
	printf("Enter the string:");
	scanf("%s",&s);
	l=strlen(s);
	for(i=0;i<l;i++)
	{
		a[i]=s[i];
	}
	printf("The copied string:%s",a);
	return 0;
}
```

5.SELECTION SORT

```
#include<stdio.h>
int main(){
int a,i,lis[10],b,j,r;
printf("Enter range:");
scanf("%d",&a);
for(i=0;i<a;i++){
	printf("Enter %d element:",i+1);
	scanf("%d",&b);
	lis[i]=b;
}
printf("\nUnsorted list is:");
for(i=0;i<a;i++){
	printf(" %d ",lis[i]);
}
for(i=0;i<a;i++){
	for(j=0;j<i;j++){
		if(lis[i]<lis[j]){
			r=lis[i];
			lis[i]=lis[j];
			lis[j]=r;
		}
	}
}
printf("\nSorted List is:");
for(i=0;i<a;i++){
	printf(" %d ",lis[i]);
}
}
```

6.REVERSE A NUMBER

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

```
int main()
{
	int n,rem,rev=0;
	printf("Enter the number:");
	scanf("%d",&n);
	while(n>0)
	{
		rem=n%10;
		rev=rev*10+rem;
		n=(n/10);
	}
	printf("The reverse of the number:%d",rev);
	return 0;

}
```

7.prime number or not

```
#include<stdio.h>

int main(){

	int n,i,count=0;

	printf("Enter the number to be checked:");

	scanf("%d",&n);

	for(i=1;i<=n;i++){

		if(n%i==0){

			count+=1;

	}}

	if(count<=2){

		printf("The given number %d is prime",n);

	}

	else{

		printf("The given number %d is not prime",n);

	}

}
```

8.**perfect number**

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

```
int main()
{
	int n,n1,sum=0,i;
	printf("Enter the number:");
	scanf("%d",&n);
	n1=n;
	for(i=1;i<n1;i++)
	{
		if(n1%i==0)
		{
			sum=sum+i;
		}
	}
	if(sum==n)
	{
		printf("%d is a Perfect number",n);
	}
	else{
		printf("%d is Not a perfect number",n);
	}
	return 0;
}
```

9.pascals triangle

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

10.palindrome

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

```
int main()
{
        char s[20];
        int l,count=0,b,i,c;
        printf("Enter the string:");
        scanf("%s",&s);
        b=strlen(s);
        c=b;
        for(i=0;i<c;i++)
        {
                if(s[i]==s[b-1])
                {
                        count=count+1;
                }
                b--;
        }
        if(count==c)
        {
                printf("%s is a palindrome",s);
        }
        else
        {
           printf("%s is not a palindrome",s);
        }
        return 0;
}
```

11.DECISION TREE

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
int calculate_cost(int freq[], int i, int j) {
   int k;
   int sum = 0;
   for (k = i; k <= j; k++) {
      sum += freq[k];
   }
   return sum;
}
void optimal_bst(int keys[], int freq[], int n) {
   int cost[n][n];
   int i, j, k, l, r;
   for (i = 0; i < n; i++) {
      cost[i][i] = freq[i];
   }
   for (l = 2; l <= n; l++) {
```

```
        for (i = 0; i <= n-l+1; i++) {
            j = i + l - 1;
            cost[i][j] = INT_MAX;
            for (k = i; k <= j; k++) {
                r = ((k > i) ? cost[i][k-1] : 0) +
                    ((k < j) ? cost[k+1][j] : 0) +
                    calculate_cost(freq, i, j);
                if (r < cost[i][j]) {
                    cost[i][j] = r;
                }
            }
        }
    }
    printf("Minimum cost = %d\n", cost[0][n-1]);
}

int main() {
    int n, i;
    printf("Enter the number of keys: ");
    scanf("%d", &n);
    int keys[n], freq[n];
    printf("Enter the keys and frequencies:\n");
    for (i = 0; i < n; i++) {
        scanf("%d %d", &keys[i], &freq[i]);
    }
    optimal_bst(keys, freq, n);
    return 0;
}
```

12.optimal cost

```
#include <limits.h>
#include <stdio.h>
#define MAX_R 10
#define MAX_C 10

int min(int x, int y, int z);
int minCost(int cost[MAX_R][MAX_C], int m, int n);

int main() {
    int cost[MAX_R][MAX_C];
    int m, n;

    printf("Enter number of rows (max %d): ", MAX_R);
    scanf("%d", &m);

    printf("Enter number of columns (max %d): ", MAX_C);
    scanf("%d", &n);
```

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

    printf("Optimal cost: %d\n", minCost(cost, m-1, n-1));
    return 0;
}

int min(int x, int y, int z) {
    if (x < y)
        return (x < z) ? x : z;
    else
        return (y < z) ? y : z;
}

int minCost(int cost[MAX_R][MAX_C], int m, int n) {
    int tc[MAX_R][MAX_C];
    tc[0][0] = cost[0][0];

    for (int i = 1; i <= m; i++)
        tc[i][0] = tc[i - 1][0] + cost[i][0];

    for (int j = 1; j <= n; j++)
        tc[0][j] = tc[0][j - 1] + cost[0][j];

    for (int i = 1; i <= m; i++)
        for (int j = 1; j <= n; j++)
            tc[i][j] = min(tc[i - 1][j - 1],
                           tc[i - 1][j],
                           tc[i][j - 1])
                     + cost[i][j];

    return tc[m][n];
}
```

13.N queens

```
#include<math.h>
int a[30],count=0;
int place(int pos)
{
int i;
for (i=1;i<pos;i++)
```

```
{
if((a[i]==a[pos])||((abs(a[i]-a[pos])==abs(i-pos))))
return 0;
}
return 1;
}
void print_sol(int n)
{
int i,j;
count++;
printf("\n\nSolution #%d:\n",count);
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
if(a[i]==j)
printf("Q\t"); else
printf("*\t");
}
printf("\n");
}
}
void queen(int n)
{
int k=1;
a[k]=0;
while(k!=0)
{
a[k]=a[k]+1;
while((a[k]<=n)&&!place(k))
a[k]++;
if(a[k]<=n)
{
if(k==n)
print_sol(n);
else
{
k++;
a[k]=0;
}
}
else
k--;
}
}
int main()
```

```
{
int i,n;
printf("Enter the number of Queens\n");
scanf("%d",&n);
queen(n);
printf("\nTotal solutions=%d",count);
return 0;
}
```

14.minimum spanning tree

```
#include <stdio.h>
#include <limits.h>
#define V 5
int minKey(int key[], bool mstSet[])
{
   int min = INT_MAX, min_index;

   for (int v = 0; v < V; v++)
      if (mstSet[v] == false && key[v] < min)
         min = key[v], min_index = v;

   return min_index;
}
void printMST(int parent[], int graph[V][V])
{
   printf("Edge \tWeight\n");
   for (int i = 1; i < V; i++)
      printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
}
void primMST(int graph[V][V])
{
   int parent[V];
   int key[V];
   bool mstSet[V];
   for (int i = 0; i < V; i++)
      key[i] = INT_MAX, mstSet[i] = false;

   key[0] = 0;
   parent[0] = -1;
   for (int count = 0; count < V - 1; count++) {
      int u = minKey(key, mstSet);
      mstSet[u] = true;
      for (int v = 0; v < V; v++)
         if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v])
            parent[v] = u, key[v] = graph[u][v];
   }
```

```
    printMST(parent, graph);
}
int main()
{
    int graph[V][V];

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

    primMST(graph);

    return 0;
}
```

15.Min max seq

```
#include <stdio.h>

int main() {
    int n, i, j;
    printf("Enter the number of integers in the list: ");
    scanf("%d", &n);

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

    printf("Minimum and Maximum value sequences for each number in the list:\n");
    for (i = 0; i < n; i++) {
        int min = arr[i], max = arr[i];
        for (j = 0; j < n; j++) {
            if (arr[j] < min) {
                min = arr[j];
            }
            if (arr[j] > max) {
                max = arr[j];
            }
        }
        printf("%d: %d %d\n", arr[i], min, max);
    }
```

```
    return 0;
}
```

16.Linear search

```
#include <stdio.h>

int main() {
    int n, i, x, a[100];
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("\nEnter the elements:\n");
    for (i = 0; i < n; i++) {
        printf("a[%d]: ", i);
        scanf("%d", &a[i]);
    }
    printf("\nEnter the element to be searched: ");
    scanf("%d", &x);
    int found = 0;
    for (i = 0; i < n; i++) {
        if (a[i] == x) {
            printf("Element is found at position %d\n", i+1);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Element not found\n");
    }
    return 0;
}
```

17.largest number in an array

```
#include<stdio.h>
int main(){
        int n,arr[10],i,max=0,b;
        printf("Enter the number of elements:");
        scanf("%d",&n);
        for(i=0;i<n;i++){
                printf("Enter %d element:",i+1);
                scanf("%d",&b);
                arr[i]=b;
                if(max<b){
                        max=b;
                }
        }
        printf("The maximum number in array is:%d",max);
```

```
}
```
18.knapsack
```
#include<stdio.h>

struct item{
   int weight, value;
};

void swap(struct item *a, struct item *b){
   struct item temp = *a;
   *a = *b;
   *b = temp;
}

void sort(struct item items[], int n){
   for(int i=0; i<n-1; i++){
      for(int j=0; j<n-i-1; j++){
         double ratio1 = (double)items[j].value / items[j].weight;
         double ratio2 = (double)items[j+1].value / items[j+1].weight;
         if(ratio1 < ratio2){
            swap(&items[j], &items[j+1]);
         }
      }
   }
}

void knapsack(struct item items[], int n, int W){
   sort(items, n);
   int weight = 0;
   double profit = 0.0;
   for(int i=0; i<n; i++){
      if(weight + items[i].weight <= W){
         weight += items[i].weight;
         profit += items[i].value;
      }
      else{
         int remaining_weight = W - weight;
         double fraction = (double)remaining_weight / items[i].weight;
         weight += remaining_weight;
         profit += fraction * items[i].value;
         break;
      }
   }
   printf("Maximum profit: %.2lf\n", profit);
}
```

```
int main(){
    int n, W;
    printf("Enter the number of items: ");
    scanf("%d", &n);
    struct item items[n];
    for(int i=0; i<n; i++){
        printf("Enter the weight and value of item %d: ", i+1);
        scanf("%d %d", &items[i].weight, &items[i].value);
    }
    printf("Enter the maximum capacity of the knapsack: ");
    scanf("%d", &W);
    knapsack(items, n, W);
    return 0;
}
```

19.GRAPH COLORING

```
#include<stdio.h>
int G[10][10],m,edges,color_tab[10],v1,v2,i,j,n,a,b;
void Gen_Col_Value(int,int);
void Gr_coloring(int,int);
int main()
{
 printf("\nEnter the number of nodes & edges\n");
 scanf("%d%d",&n,&edges);
 m=n-1;
 printf("\nEnter the edges of the graph\n\n");
 for (i=1;i<=edges; i++)
 {
  printf("Enter value of x,y\n");
  scanf("%d%d",&v1,&v2);
   G[v1][v2] = G[v2][v1] = 1;
   printf("\n");
 }
 Gr_coloring(1,n);
 printf("\n The Vertices To be Coloured As...\n");
  for(i=1;i<=n;i++)
    printf("\n V%d:=%d",i,color_tab[i]);
 return 0;
}
void Gen_Col_Value(int k,int n)
{
 while(1)
 {
  a=color_tab[k]+1;
  b=m+1;
  color_tab[k] = a%b;
```

```
  if(color_tab[k]==0) return;
  for(j=1;j<=n;j++)
  {
   if(G[k][j] && color_tab[k]==color_tab[j])
     break;
  }
  if(j==n+1) return;
   }
}
void Gr_coloring(int k,int n)
{
  Gen_Col_Value(k,n);
  if(color_tab[k]==0) return;
  if(k==n)  return;
  else Gr_coloring(k+1,n);
}
```

20.Insert elements in an array

```
#include <stdio.h>
int main()
{
   int arr1[100],i,n,p,inval;
      printf("\n\nInsert New value in the sorted array :\n");
    printf("Input the size of array : ");
    scanf("%d", &n);
      printf("Input %d elements in the array in ascending order:\n",n);
      for(i=0;i<n;i++)
           {
                printf("element - %d : ",i);
                scanf("%d",&arr1[i]);
              }
   printf("Input the value to be inserted : ");
   scanf("%d",&inval);
   printf("The exist array list is :\n");
   for(i=0;i<n;i++)
     printf("% 5d",arr1[i]);
   for(i=0;i<n;i++)
   {

    if(inval<arr1[i])
     {
       p = i;
       break;
     }
     else
```

```
    {
        p=i+1;
    }
        }
  for(i=n+1;i>=p;i--)
    arr1[i]= arr1[i-1];
    arr1[p]=inval;
    printf("\n\nAfter Insert the list is :\n");
  for(i=0;i<=n;i++)
    printf("% 5d",arr1[i]);
        printf("\n");
}
```

21.Hamilton circuit

```
#include<iostream>

#define NODE 4

using namespace std;

int graph[NODE][NODE];

int path[NODE];

void displayCycle() {

  cout<<"The Hamilton Cycle : " << endl;

  for (int i = 0; i < NODE; i++)

    cout << path[i] << "  ";

  cout << path[0] << endl;

}

bool isValid(int v, int k) {

  if (graph [path[k-1]][v] == 0)

    return false;

  for (int i = 0; i < k; i++)

    if (path[i] == v)
```

```
      return false;
   return true;
}
bool cycleFound(int k) {
   if (k == NODE) {
      if (graph[path[k-1]][ path[0] ] == 1 )
         return true;
      else
         return false;
   }
   for (int v = 1; v < NODE; v++) {
      if (isValid(v,k)) {
         path[k] = v;
         if (cycleFound (k+1) == true)
            return true;
         path[k] = -1;
      }
   }
   return false;
}
bool hamiltonianCycle() {
   for (int i = 0; i < NODE; i++)
      path[i] = -1;
```

```
   path[0] = 0;
   if ( cycleFound(1) == false ) {
      cout << "Solution does not exist"<<endl;
      return false;
   }
   displayCycle();
   return true;
}
int main() {
        int i,j;  cout << "Enter the Graph : " << endl;
        for (i=0;i<NODE;i++){
                for (j=0;j<NODE;j++){
                        cout << "Graph G(" << (i+1) << "," << (j+1) << ") = ";
                        cin >> graph[i][j];
                }
        }
        cout << endl;
        cout << "The Graph :" << endl;
        for (i=0;i<NODE;i++){
                for (j=0;j<NODE;j++){
                        cout << graph [i][j] << "\t";
                }
                cout << endl;
```

```
        }


        cout << endl;


        hamiltonianCycle();

  }
```

22.Generation of prime

```
#include<stdio.h>
int main()
{
        int n,i,j,c;
        printf("Enter the range:");
        scanf("%d",&n);
        for(i=2;i<=n;i++)
        {
                c=0;
                for(j=1;j<=i;j++){
                        if(i%j==0)
                        {
                                c=c+1;
                        }
                }
                if(c<=2)
           {
                 printf("%d\n",i);
           }
        }
        return 0;


}
```

23.GCD

```
#include<stdio.h>
int main(){
        int n1,n2,i,gcd=0;
        printf("Enter 1st number:");
        scanf("%d",&n1);
        printf("Enter 2nd number:");
        scanf("%d",&n2);
        for(i=1;i<=n1 && i<=n2;i++){
```

```
                if(n1%i==0 && n2%i==0){
                        gcd=i;
                }
}
printf("\nThe Gcd of 2 Numbers is:%d",gcd);
printf("\nThe lcm of 2 Numbers is:%d",((n1*n2)/gcd));
}
```

24.floyds algorithm

```
#include<stdio.h>
#include<cstdlib>
void floydWarshall(int **graph, int n)
{
   int i, j, k;
   for (k = 0; k < n; k++)
   {
      for (i = 0; i < n; i++)
      {
         for (j = 0; j < n; j++)
         {
            if (graph[i][j] > graph[i][k] + graph[k][j])
               graph[i][j] = graph[i][k] + graph[k][j];
         }
      }
   }
}

int main(void)
{
   int n, i, j;
   printf("Enter the number of vertices: ");
   scanf("%d", &n);
   int **graph = (int **)malloc((long unsigned) n * sizeof(int *));
   for (i = 0; i < n; i++)
   {
      graph[i] = (int *)malloc((long unsigned) n * sizeof(int));
   }
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < n; j++)
      {
         if (i == j)
            graph[i][j] = 0;
         else
            graph[i][j] = 100;
      }
   }
```

```
    printf("Enter the edges: \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("[%d][%d]: ", i, j);
            scanf("%d", &graph[i][j]);
        }
    }
    printf("The original graph is:\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("%d ", graph[i][j]);
        }
        printf("\n");
    }
    floydWarshall(graph, n);
    printf("The shortest path matrix is:\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("%d ", graph[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

25.Fibonacci series

```
#include<stdio.h>
int fib(int a){
        if(a==0){
                return 0;
        }
        else if(a==1){
                return 1;
        }
        else{
                return fib(a-1)+fib(a-2);
        }
}
int main(){
        int n,i;
```

```
	printf("Enter the range:");
	scanf("%d",&n);
	for(i=0;i<n;i++){
		printf(" %d ",fib(i));
	}
}
```

26.factorial

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

27.number of factors

```
#include<stdio.h>
int main()
{
	int n,i,count=0;
	printf("Enter the number:");
	scanf("%d",&n);
	for(i=1;i<=n;i++)
	{
		if(n%i==0)
		{
			printf("%d\n",i);
			count++;
		}
	}
	printf("The no.of.factors:%d",count);

	return 0;
}
```

28.container loading problem

```
#include <stdio.h>

#define MAX_CONTAINERS 100

#define MAX_WEIGHT 1000
```

```
int bestLoading[MAX_CONTAINERS];
int currentLoading[MAX_CONTAINERS];
int containers[MAX_CONTAINERS];
int numContainers;
int maxWeight;
int bestWeight;
void containerLoader(int level, int currentWeight) {
    if (level == numContainers) {
        if (currentWeight > bestWeight) {
            bestWeight = currentWeight;
            for (int i = 0; i < numContainers; i++) {
                bestLoading[i] = currentLoading[i];
            }
        }
        return;
    }
    currentLoading[level] = 0;
    containerLoader(level + 1, currentWeight);
    if (currentWeight + containers[level] <= maxWeight) {
        currentLoading[level] = 1;
        containerLoader(level + 1, currentWeight + containers[level]);
    }
}
```

```
int main() {
    printf("Enter the number of containers: ");
    scanf("%d", &numContainers);
    printf("Enter the maximum weight capacity: ");
    scanf("%d", &maxWeight);
    printf("Enter the weight of each container:\n");
    for (int i = 0; i < numContainers; i++) {
        scanf("%d", &containers[i]);
    }
    bestWeight = 0;
    containerLoader(0, 0);
    printf("Best loading configuration:\n");
    for (int i = 0; i < numContainers; i++) {
        if (bestLoading[i]) {
            printf("Container %d\n", i + 1);
        }
    }
    printf("Total weight: %d\n", bestWeight);
    return 0;
}
```

29.BINOMIAL COEFFICIENT

```
#include<stdio.h>
int fact(int x)
{
        int fact=1;
        while(x!=0)
```

```
        {
                fact=fact*x;
                x--;
        }
        return fact;

}
int main()
{
        int n,c,g;
        printf("Enter the value of n:");
        scanf("%d",&n);
        printf("Enter the value of c:");
        scanf("%d",&c);
        g=fact(n)/(fact(c)*fact(n-c));
        printf("The Bionomial co-efficient:%d",g);
        return 0;

}
```

30.ASSIGNMENT PROBLEM

```
#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX_WORKERS 10

#define MAX_TASKS 10

int numWorkers, numTasks;

int costMatrix[MAX_WORKERS][MAX_TASKS];

bool assigned[MAX_WORKERS], visited[MAX_TASKS];

int assignment[MAX_WORKERS];

int minCost = 0x7FFFFFFF;  // Initialize with a large value

void inputCostMatrix() {

   printf("Enter the cost matrix (%d x %d):\n", numWorkers, numTasks);

   for (int i = 0; i < numWorkers; i++) {
```

```
        for (int j = 0; j < numTasks; j++) {
            scanf("%d", &costMatrix[i][j]);
        }
    }
}
int calculateCost() {
    int totalCost = 0;
    for (int i = 0; i < numWorkers; i++) {
        totalCost += costMatrix[i][assignment[i]];
    }
    return totalCost;
}
void branchAndBound(int worker, int cost) {
    if (worker == numWorkers) {
        if (cost < minCost) {
            minCost = cost;
        }
        return;
    }
    for (int task = 0; task < numTasks; task++) {
        if (!visited[task] && cost + costMatrix[worker][task] < minCost) {
            visited[task] = true;
            assignment[worker] = task;
```

```
            branchAndBound(worker + 1, cost + costMatrix[worker][task]);
            visited[task] = false;
        }
    }
}
void printAssignment() {
    printf("Optimal assignment:\n");
    for (int i = 0; i < numWorkers; i++) {
        printf("Worker %d -> Task %d\n", i + 1, assignment[i] + 1);
    }
}
int main() {
    printf("Enter the number of workers: ");
    scanf("%d", &numWorkers);
    printf("Enter the number of tasks: ");
    scanf("%d", &numTasks);
    if (numWorkers > MAX_WORKERS || numTasks > MAX_TASKS) {
        printf("Exceeded maximum limit of workers or tasks.\n");
        return 0;
    }
    inputCostMatrix();
    branchAndBound(0, 0);
    printf("Minimum cost: %d\n", minCost);
```

```
    printAssignment();

    return 0;

}
```

31.ARMSTRONG NUMBER

```
#include<stdio.h>
#include<math.h>
int main(){
        int n,t,rem,sum=0,count,r;
        printf("Enter the numbet to check:");
        scanf("%d",&n);
        t=n;
        r=n;
        count=(n==0)?1:log10(n)+1;
        while(t>0){
                rem=t%10;
                sum+=pow(rem,count);
                t=(t/10);
        }
        if(n==sum){
                printf("The given number %d is armstrong",n);
        }
        else{
                printf("The given number %d is not a armstrong",n);
        }
}
```

32.BUBBLE SORT

```
#include<stdio.h>
void bubble_sort(long[],long);
int main()
{
        long array[100],n,c;
        printf("Enter number of elements\n");
        scanf("%d,&n");
        printf("Enter %ld integers\n",n);
        for(c=0;c<n;c++)
        scanf("%d",&array[c]);
        bubble_sort(array,n);
        printf("Sorted list in ascending order:\n");
        for(c=0;c<n;c++)
        printf("%ld\n",array[c]);
        return 0;
}
void bubble_sort(long list[],long n)
```

```
{
        long c,d,t;
        for(c=0;c<n-1;c++)
        {
                for(d=0;d<n-c-1;d++)
                {
                        if(list[d]>list[d+1])
                        {
                                t=list[d];
                                list[d]=list[d+1];
                                list[d+1]=t;
                        }
                }
        }
}
```

33.MATRIX MULTIPLICATION

```
#include<stdio.h>
int main()
{
        int a[10][10],b[10][10],i,j,k,res[10][10],n1,n2,n3,n4;
        printf("Enter the rows of matrix 1:");
        scanf("%d",&n1);
        printf("Enter the columns of matrix 1:");
        scanf("%d",&n2);
        printf("Enter the rows of matrix 2:");
        scanf("%d",&n3);
        printf("Enter the coloumns of matrix 2:");
        scanf("%d",&n4);
        if(n2!=n3)
        {
                printf("Matrix multiplication is not possible!!");
        }
        else
        {
                printf("Enter the elements of matrix 1:\n");
                for(i=0;i<n1;i++)
                {
                        for(j=0;j<n2;j++)
                        {
                                printf("a[%d][%d]:",i+1,j+1);
                                scanf("%d",&a[i][j]);
                        }
                }
                printf("Enter the elements of matrix 2:\n");
                for(i=0;i<n3;i++)
                {
```

```
                for(j=0;j<n4;j++)
                {
                        printf("b[%d][%d]:",i+1,j+1);
                        scanf("%d",&b[i][j]);
                }
        }
        for(i=0;i<n1;i++)
        {
                for(j=0;j<n3;j++)
                {
                        res[i][j]=0;
                        for(k=0;k<n2;k++)
                        {
                                res[i][j]+=a[i][k]*b[k][j];
                        }
                }
        }
        printf("The product for the matrices are:\n");
        for(i=0;i<n1;i++)
        {
                for(j=0;j<n4;j++)
                {
                        printf("%d\t",res[i][j]);
                }
                printf("\n");
        }
    }
    return 0;

}
```

34.copy a string

```
#include<stdio.h>
#include<string.h>
int main()
{
    char s[20],a[20];
    int l,i;
    printf("Enter the string:");
    scanf("%s",&s);
    l=strlen(s);
    for(i=0;i<l;i++)
    {
        a[i]=s[i];
    }
    printf("The copied string:%s",a);
```

```
	return 0;
}
```

35.Binary search

```
#include<stdio.h>
int main()
{
	int a[20],i,j,m,n,temp,s,mid,low,high,pos=0,o=0;
	printf("Enter the number of elements:");
	scanf("%d",&n);
	printf("Enter the elements:\n");
	for(i=0;i<n;i++)
	{
		printf("Enter the element %d:",i+1);
		scanf("%d",&a[i]);
	}
	for(i=0;i<n;i++)
	{
		for(j=i;j<n;j++)
		{
			if(a[i]>a[j])
			{
				temp=a[i];
				a[i]=a[j];
				a[j]=temp;
			}

		}
	}
	printf("The sorted order\n");
	for(i=0;i<n;i++)
	{
		printf("%d\t",a[i]);
	}
	printf("\nEnter the element to be searched:");
	scanf("%d",&s);
	high=n;
	low=0;
	while(pos==0 && o<=n)
	{
		mid=(low+high)/2;

		if(a[mid]==s)
		{
		   pos=mid+1;
		}
```

```
                else if(a[mid]<s)
                {
                        low=mid+1;

                }
                else
                {
                        high=mid-1;

                }
                o++;
        }
        if(pos!=0)
        {
                printf("The position of the given element %d is:%d",s,pos);
        }
        else
        {
          printf("element not found");
   }
        return 0;
}
```

36.Reverse of a string

```
#include<stdio.h>
#include<string.h>
int main()
{
        char s[20],d[20];
        int count=0,b,i,c;
        printf("Enter the string:");
        scanf("%s",&s);
        b=strlen(s);
        c=b;
        for(i=0;i<b;i++)
        {
                d[i]=s[c-1];
                c--;
        }
        printf("The reverse of the string is:%s",d);
        return 0;
}
```

37.Length of the string

```
#include<stdio.h>
#include<string.h>
int main()
{
```

```
        char s[20];
        int count=0,i=0;
        printf("Enter the string:");
        scanf("%s",&s);
        while(s[i]!='\0')
        {
                count=count+1;
                i++;
        }
        printf("The length of the string:%d",count);
}
```

38.Merge sort

```
#include <stdio.h>
void merge_sort(int i, int j, int a[], int aux[])
{
   if (j <= i)
         {
      return;
   }
   int mid = (i + j) / 2;
   merge_sort(i, mid, a, aux);
         merge_sort(mid + 1, j, a, aux);

   int pointer_left = i;
   int pointer_right = mid + 1;
   int k;

   for (k = i; k <= j; k++) {
      if (pointer_left == mid + 1) {
         aux[k] = a[pointer_right];
         pointer_right++;
      } else if (pointer_right == j + 1) {
         aux[k] = a[pointer_left];
         pointer_left++;
      } else if (a[pointer_left] < a[pointer_right]) {
         aux[k] = a[pointer_left];
         pointer_left++;
      } else {
         aux[k] = a[pointer_right];
         pointer_right++;
      }
   }

   for (k = i; k <= j; k++) {
      a[k] = aux[k];
```

```
    }
}
int main()
 {
  int a[100], aux[100], n, i, d, swap;

  printf("Enter number of elements in the array:\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

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

  merge_sort(0, n - 1, a, aux);

  printf("Printing the sorted array:\n");

  for (i = 0; i < n; i++)
     printf("%d\n", a[i]);

  return 0;
}
```

39.Strassens 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 elements of first matrix: ");
  for(i = 0;i < 2; i++)
      for(j = 0;j < 2; j++)
          scanf("%d", &a[i][j]);

  printf("Enter the elements of second matrix: ");
  for(i = 0; i < 2; i++)
      for(j = 0;j < 2; j++)
          scanf("%d", &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("\nMultiplication 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;
}
```