DAA-CSA0626

Lab programs

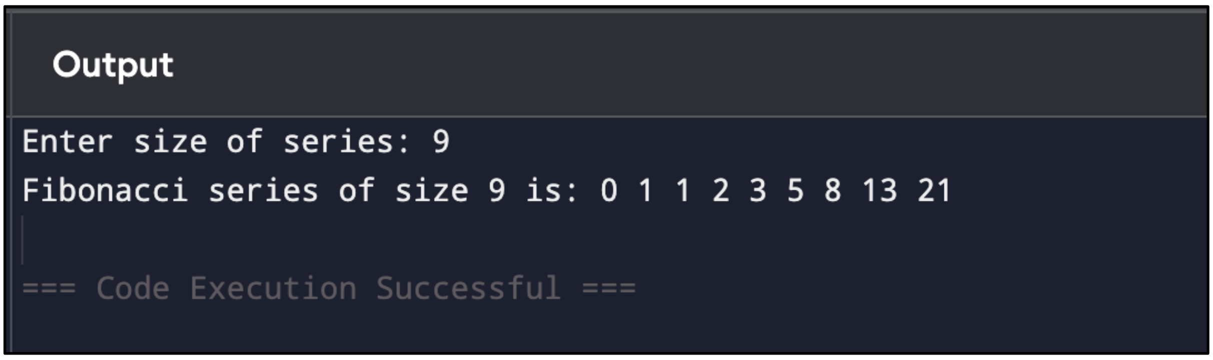
192210202

1.Fibonacci series using recursion

Code:

```
#include <stdio.h>
int fibo(int n) {
   if (n == 0)
      return 0;
   else if (n == 1)
      return 1;
   else
      return fibo(n - 1) + fibo(n - 2);
}
int main() {
   int n, i;
   printf("Enter size of series: ");
   scanf("%d", &n);
   printf("Fibonacci series of size %d is: ", n);
   for (i = 0; i < n; i++) {
      printf("%d ", fibo(i));
   }
   return 0;
}
```

Output:



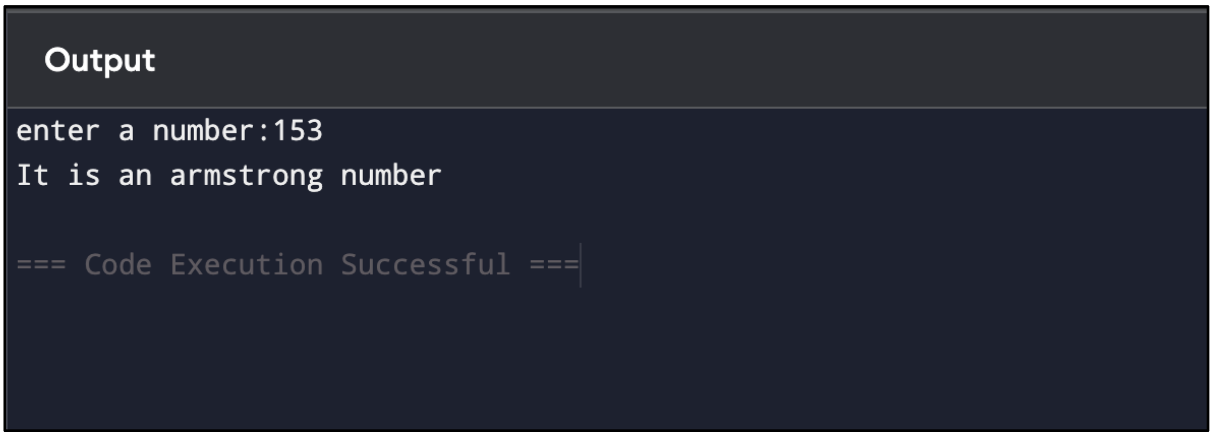
```
Enter size of series: 9
Fibonacci series of size 9 is: 0 1 1 2 3 5 8 13 21

=== Code Execution Successful ===
```

2.Armstrong number

Code:

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

Output:



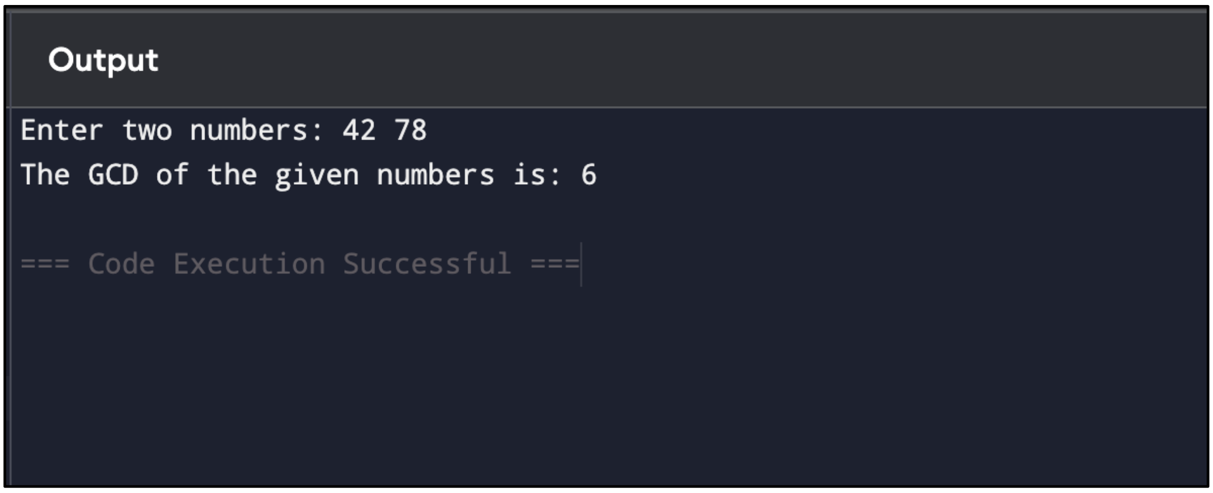
```
enter a number:153
It is an armstrong number

=== Code Execution Successful ===
```

3.GCD of numbers

Code:

```
#include <stdio.h>
int main() {
    int a, b, temp;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    while (b != 0) {
        temp = b;
        b = a % b;
        a = temp;
    }
    printf("The GCD of the given numbers is: %d", a);
    return 0;
}
```

Output:



Output

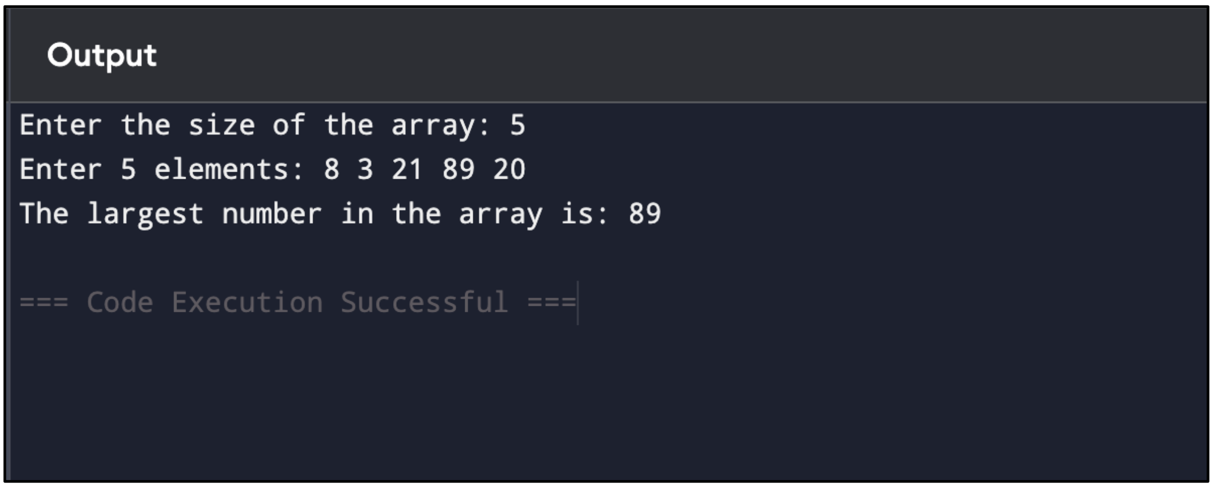
```
Enter two numbers: 42 78
The GCD of the given numbers is: 6

=== Code Execution Successful ===
```

4.Largest number in array

Code:

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

Output:



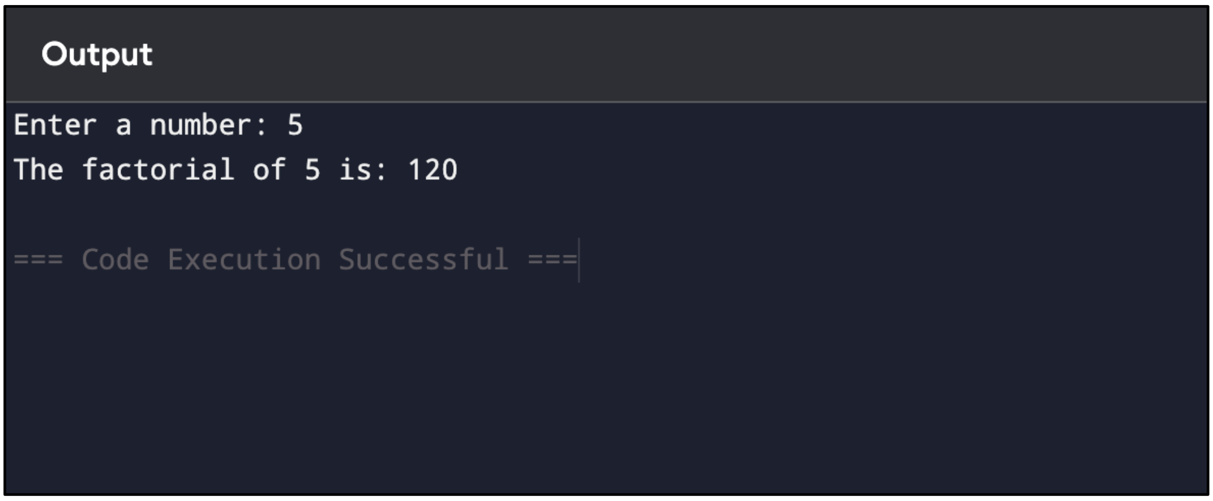
```
Enter the size of the array: 5
Enter 5 elements: 8 3 21 89 20
The largest number in the array is: 89

=== Code Execution Successful ===
```

5.Factorial of a number using recursion

Code:

```
#include <stdio.h>
int factorial(int n) {
    if (n==0||n==1)
        return 1;
    return n*factorial(n-1);
}
int main(){
    int n;
    printf("Enter a number: ");
    scanf("%d",&n);
    printf("The factorial of %d is: %d",n,factorial(n));
    return 0;
}
```

Output:


Output

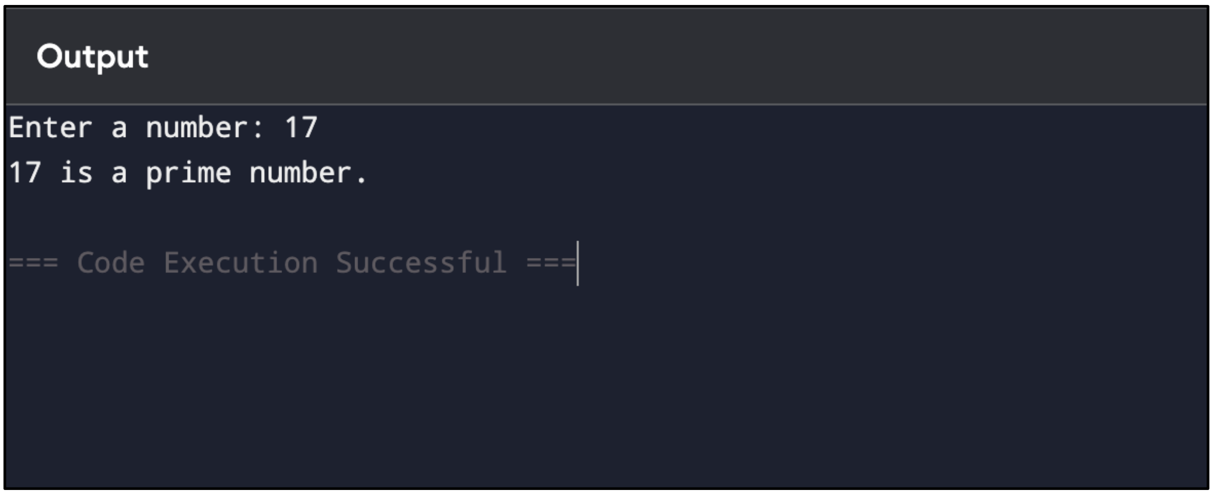
```
Enter a number: 5
The factorial of 5 is: 120


=== Code Execution Successful ===
```

6.Prime number

Code:

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

Output:

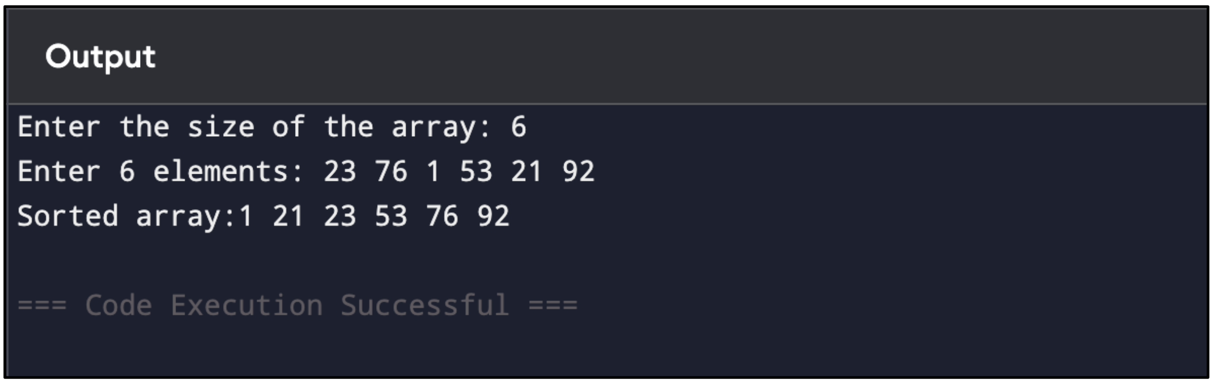


```
Enter a number: 17
17 is a prime number.

=== Code Execution Successful ===
```

7.Selection Sort

Code:

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



Output

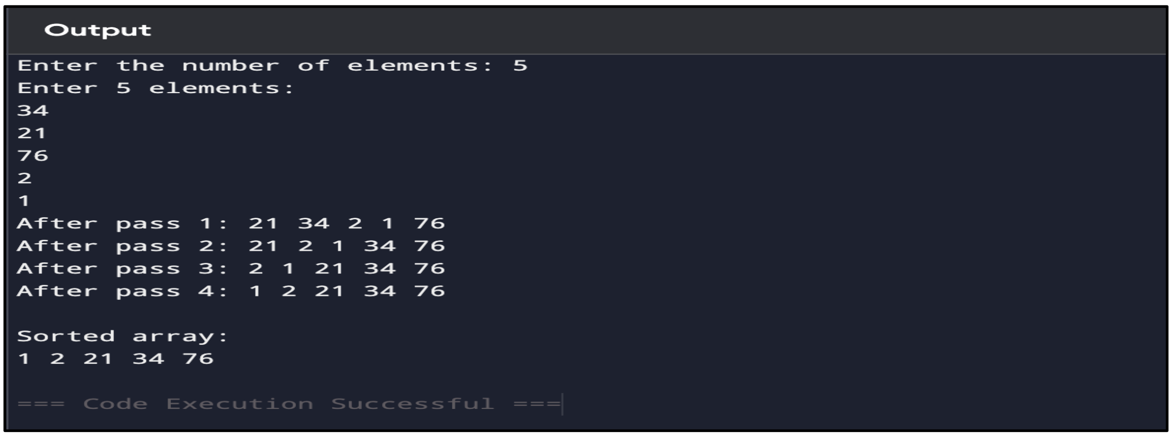
```
Enter the size of the array: 6
Enter 6 elements: 23 76 1 53 21 92
Sorted array:1 21 23 53 76 92


=== Code Execution Successful ===
```

8.Bubble Sort

Code:

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

Output:



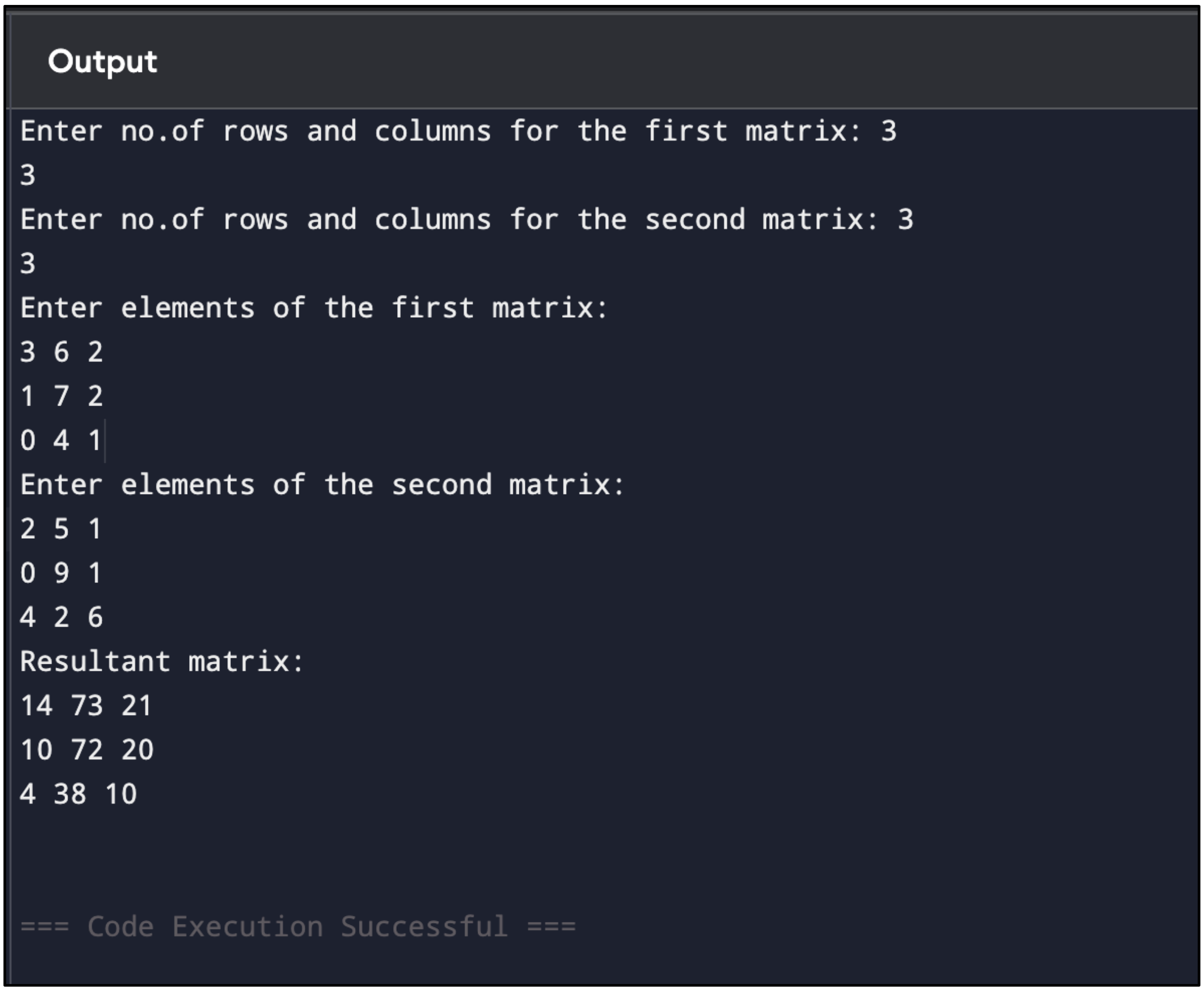
```
Output
Enter the number of elements: 5
Enter 5 elements:
34
21
76
2
1
After pass 1: 21 34 2 1 76
After pass 2: 21 2 1 34 76
After pass 3: 2 1 21 34 76
After pass 4: 1 2 21 34 76

Sorted array:
1 2 21 34 76

=== Code Execution Successful ===
```

9.Matrix Multiplication

Code:

```
#include<stdio.h>
int main(){
    int a[10][10],b[10][10],result[10][10],r1,c1,r2,c2,i,j,k;
    printf("Enter no.of rows and columns for the first matrix: ");
    scanf("%d%d",&r1,&c1);
    printf("Enter no.of rows and columns for the second matrix: ");
    scanf("%d%d",&r2,&c2);
    if(c1!=r2){
        printf("Matrix multiplication not possible.\n");
        return 0;
    }
    printf("Enter elements of the first matrix:\n");
    for(i=0;i<r1;i++){
        for(j=0;j<c1;j++){
            scanf("%d",&a[i][j]);
        }
    }
    printf("Enter elements of the second matrix:\n");
    for(i=0;i<r2;i++){
        for(j=0;j<c2;j++){
            scanf("%d",&b[i][j]);
        }
    }
    for(i=0;i<r1;i++){
        for(j=0;j<c2;j++){
            result[i][j]=0;
            for(k=0;k<c1;k++){
                result[i][j]+=a[i][k]*b[k][j];
            }
        }
    }
    printf("Resultant matrix:\n");
    for(i=0;i<r1;i++){
        for(j=0;j<c2;j++){
            printf("%d ",result[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:



```
Enter no.of rows and columns for the first matrix: 3
3
Enter no.of rows and columns for the second matrix: 3
3
Enter elements of the first matrix:
3 6 2
1 7 2
0 4 1
Enter elements of the second matrix:
2 5 1
0 9 1
4 2 6
Resultant matrix:
14 73 21
10 72 20
4 38 10


=== Code Execution Successful ===
```

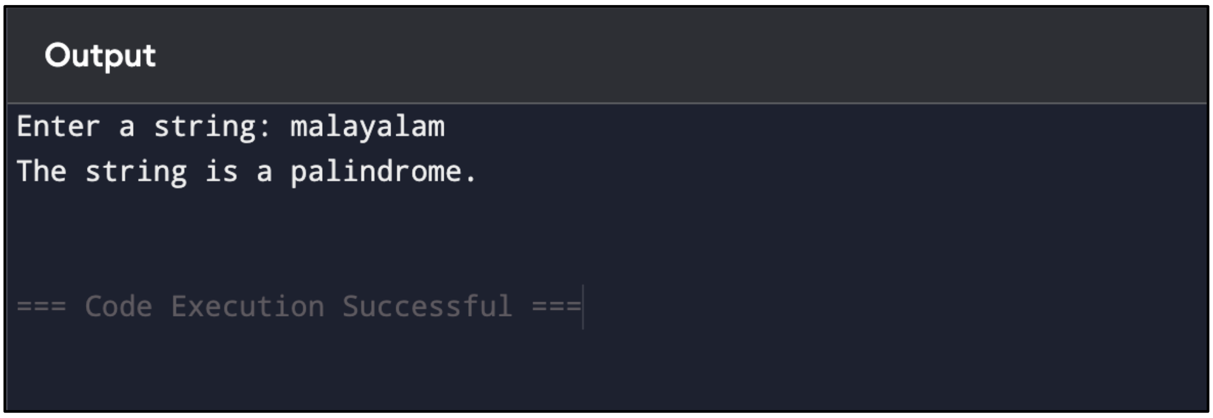
10.String Palindrome

Code:

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

```
int main(){
    char str[100];
    int i,len,flag=1;
    printf("Enter a string: ");
    scanf("%s",str);
    len=strlen(str);
    for(i=0;i<len/2;i++){
        if(str[i]!=str[len-i-1]){
            flag=0;
            break;
        }
    }
    if(flag){
        printf("The string is a palindrome.\n");
    }
    else{
    printf("The string is not a palindrome.\n");
    }
    return 0;
}
```

Output:



```
Enter a string: malayalam
The string is a palindrome.


=== Code Execution Successful ===
```

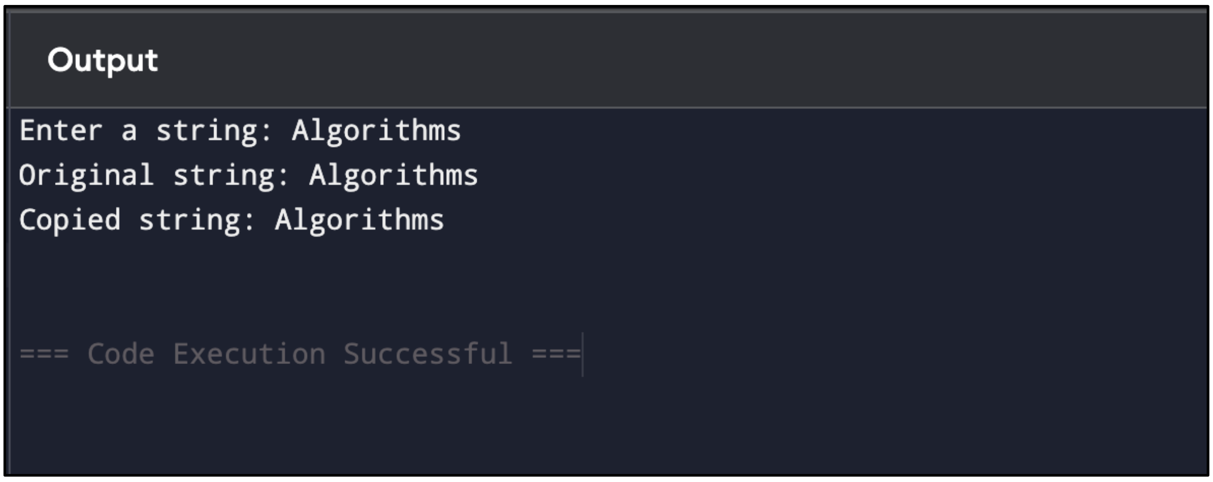
11.String Copy

Code:

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

```
int main(){
    char str1[100],str2[100];
    int i;
    printf("Enter a string: ");
    scanf("%s",str1);
    for(i=0;str1[i]!='\0';i++){
        str2[i]=str1[i];
    }
    str2[i]='\0';
    printf("Original string: %s\n",str1);
    printf("Copied string: %s\n",str2);
    return 0;
}
```

Output:



Output

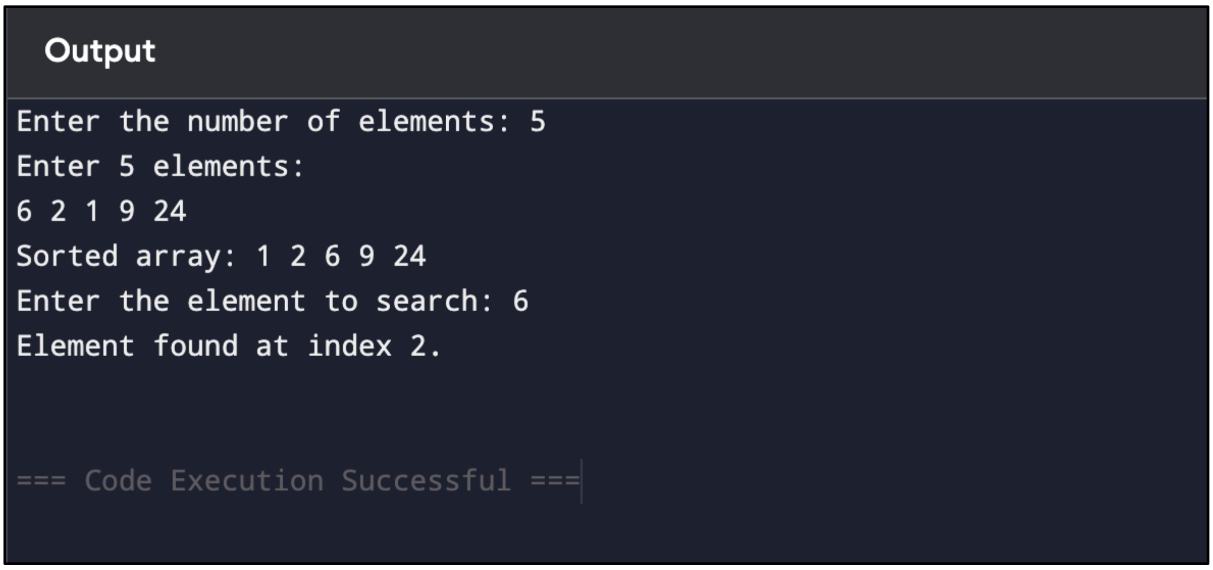
```
Enter a string: Algorithms
Original string: Algorithms
Copied string: Algorithms


=== Code Execution Successful ===
```

12.Binary Search

Code:

```
#include<stdio.h>
int main(){
    int arr[100],n,i,j,temp,key,low,high,mid;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("Enter %d elements:\n",n);
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    for(i=0;i<n;i++){
        for(j=i+1;j<n;j++){
            if(arr[i]>arr[j]){
                temp=arr[i];
                arr[i]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }
    printf("Sorted array: ");
    for(i=0;i<n;i++){
        printf("%d ",arr[i]);
    }
    printf("\n");
    printf("Enter the element to search: ");
    scanf("%d",&key);
    low=0;
    high=n-1;
    while(low<=high){
        mid=(low+high)/2;
        if(arr[mid]==key){
            printf("Element found at index %d.\n",mid);
            return 0;
        }
        else if(arr[mid]<key){
            low=mid+1;
        }
        else{
            high=mid-1;
        }
    }
    printf("Element not found.\n");
    return 0;
}
```

Output:



Output

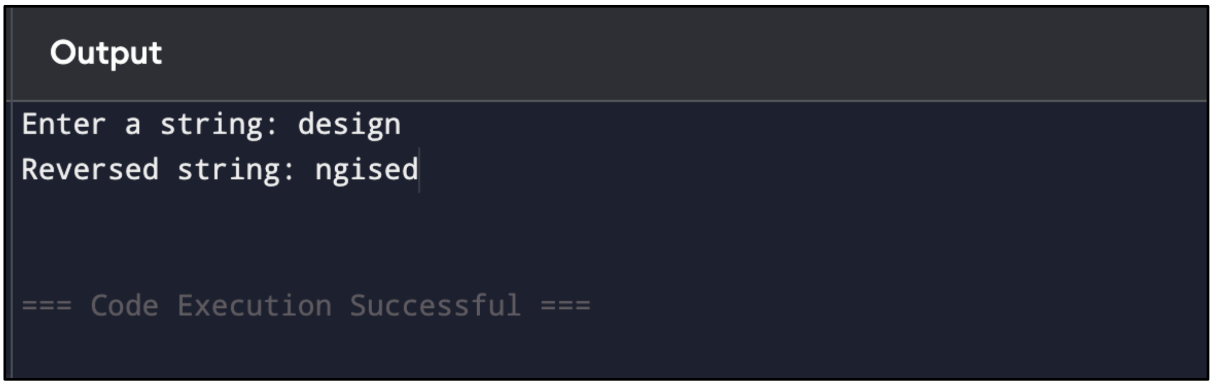
```
Enter the number of elements: 5
Enter 5 elements:
6 2 1 9 24
Sorted array: 1 2 6 9 24
Enter the element to search: 6
Element found at index 2.


=== Code Execution Successful ===
```

13.String Reverse

Code:

```
#include<stdio.h>
int main(){
   char str[100],rev[100];
   int i,len=0;
   printf("Enter a string: ");
   scanf("%s",str);
   while(str[len]!='\0'){
      len++;
   }
   for(i=0;i<len;i++){
      rev[i]=str[len-i-1];
   }
   rev[len]='\0';
   printf("Reversed string: %s\n",rev);
   return 0;
}
```

Output:



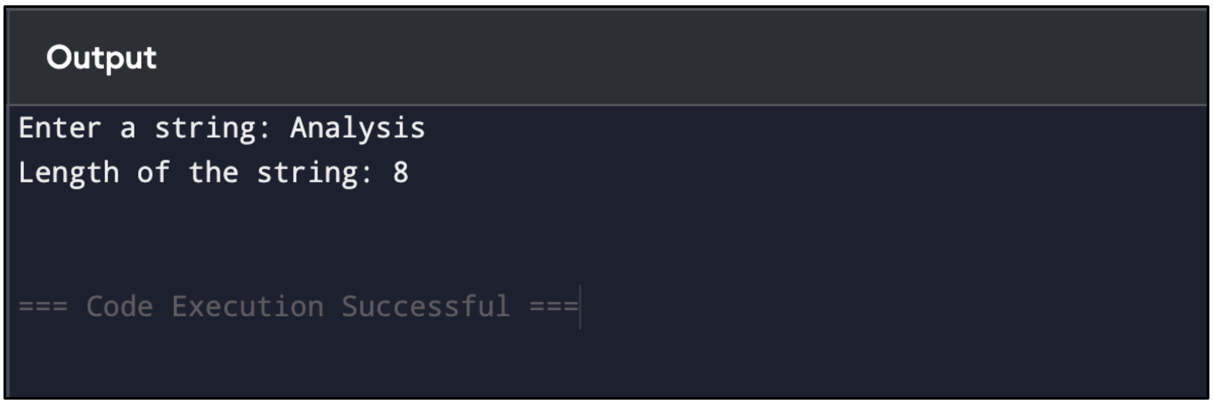
```
Enter a string: design
Reversed string: ngised


=== Code Execution Successful ===
```

14.Length of String

Code:

```
#include<stdio.h>
int main(){
   char str[100];
   int i,length=0;
   printf("Enter a string: ");
   scanf("%s",str);
   for(i=0;str[i]!='\0';i++){
      length++;
   }
   printf("Length of the string: %d\n",length);
   return 0;
}
```

Output:



```
Enter a string: Analysis
Length of the string: 8


=== Code Execution Successful ===
```

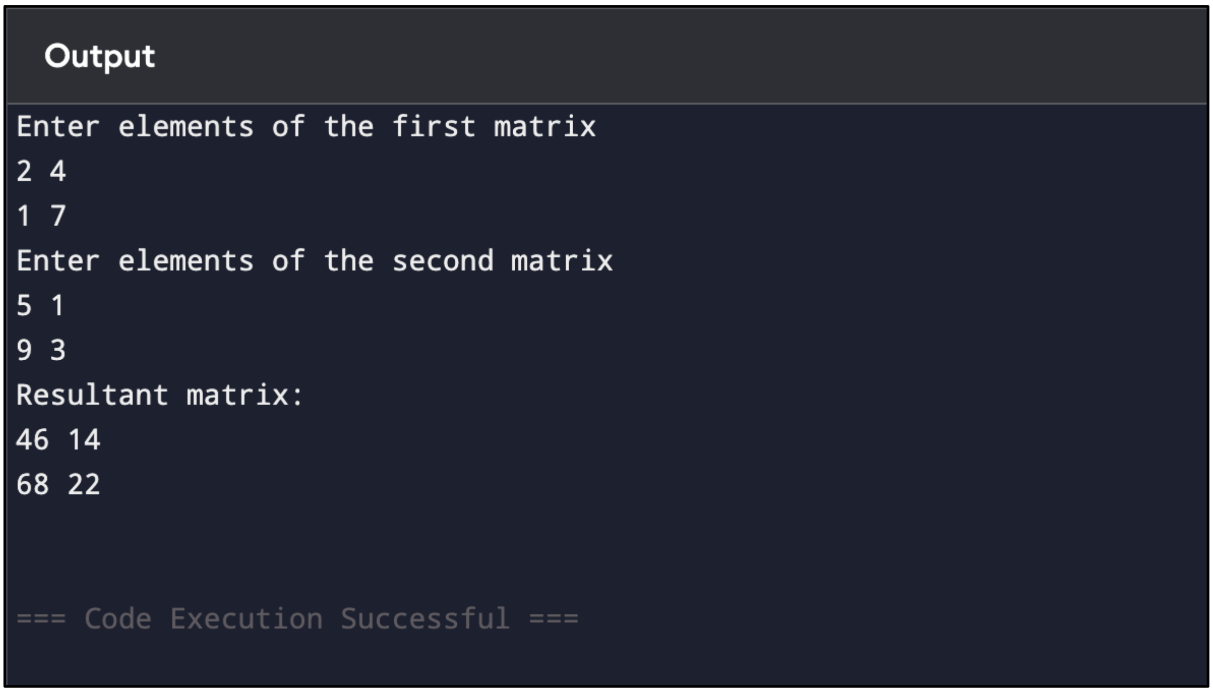
15.Strassen's Matrix Multiplication

Code:

```
#include<stdio.h>
void st(int A[2][2], int B[2][2], int C[2][2]) {
   int P,Q,R,S,T,U,V;
   P=(A[0][0]+A[1][1])*(B[0][0]+B[1][1]);
   Q=(A[1][0]+A[1][1])*B[0][0];
   R=A[0][0]*(B[0][1]-B[1][1]);
   S=A[1][1]*(B[1][0]-B[0][0]);
   T=(A[0][0]+A[0][1])*B[1][1];
   U=(A[1][0]-A[0][0])*(B[0][0]+B[0][1]);
   V=(A[0][1]-A[1][1])*(B[1][0]+B[1][1]);
   C[0][0]=P+S-T+V;
   C[0][1]=R+T;
   C[1][0]=Q+S;
   C[1][1]=P+R-Q+U;
}
int main() {
   int A[2][2],B[2][2],C[2][2];
   printf("Enter elements of the first matrix\n");
   for (int i=0;i<2;i++) {
      for (int j=0;j<2;j++) {
         scanf("%d",&A[i][j]);
      }
   }
   printf("Enter elements of the second matrix\n");
   for (int i=0;i<2;i++) {
      for (int j=0;j<2;j++) {
         scanf("%d",&B[i][j]);
      }
   }
  st(A,B,C);
  printf("Resultant matrix:\n");
   for(int i=0;i<2;i++){
      for(int j=0;j<2;j++){
         printf("%d ", C[i][j]);
```

```
        }
        printf("\n");
    }
    return 0;
}
```

Output:



Output

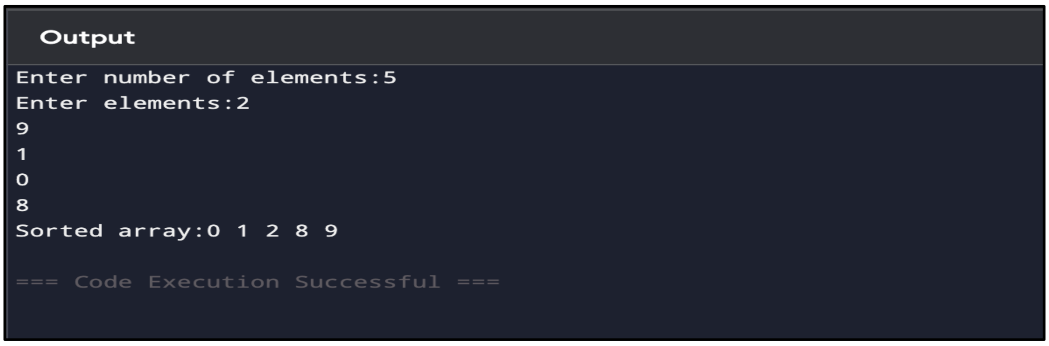
```
Enter elements of the first matrix
2 4
1 7
Enter elements of the second matrix
5 1
9 3
Resultant matrix:
46 14
68 22


=== Code Execution Successful ===
```

16.MERGE SORT

CODE:

```
#include<stdio.h>
int main(){
    int n;
    printf("Enter number of elements:");
    scanf("%d",&n);
    int arr[n],temp[n];
    printf("Enter elements:");
    for(int i=0;i<n;i++)scanf("%d",&arr[i]);
    for(int size=1;size<n;size*=2){
        for(int left=0;left<n-1;left+=2*size){
            int mid=left+size-1;
            int right=(left+2*size-1<n-1)?left+2*size-1:n-1;
            int i=left,j=mid+1,k=left;
            while(i<=mid&&j<=right){
                if(arr[i]<=arr[j])temp[k++]=arr[i++];
                else temp[k++]=arr[j++];
            }
            while(i<=mid)temp[k++]=arr[i++];
            while(j<=right)temp[k++]=arr[j++];
        }
        for(int i=0;i<n;i++)arr[i]=temp[i];
    }
    printf("Sorted array:");
    for(int i=0;i<n;i++)printf("%d ",arr[i]);
    return 0;
}
```

OUTPUT:

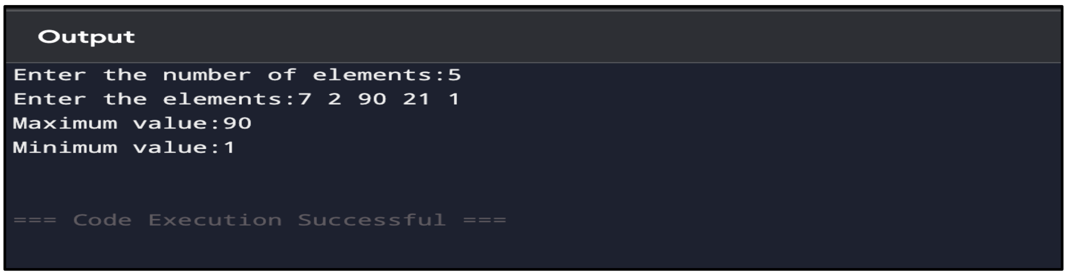

```
Output
Enter number of elements:5
Enter elements:2
9
1
0
8
Sorted array:0 1 2 8 9

=== Code Execution Successful ===
```

17.MIN and MAX

CODE:

```
#include<stdio.h>
int main(){
   int n;
   printf("Enter the number of elements:");
   scanf("%d",&n);
   int arr[n];
   printf("Enter the elements:");
   for(int i=0;i<n;i++)scanf("%d",&arr[i]);
   int max,min,mid,low=0,high=n-1,max1,min1,max2,min2;
   while(low<high){
      mid=(low+high)/2;
      max1=arr[low],min1=arr[low];
      for(int i=low;i<=mid;i++){
         if(arr[i]>max1)max1=arr[i];
         if(arr[i]<min1)min1=arr[i];
      }
      max2=arr[mid+1],min2=arr[mid+1];
      for(int i=mid+1;i<=high;i++){
         if(arr[i]>max2)max2=arr[i];
         if(arr[i]<min2)min2=arr[i];
      }
      max=(max1>max2)?max1:max2;
      min=(min1<min2)?min1:min2;
      break;
   }
   printf("Maximum value:%d\n",max);
   printf("Minimum value:%d\n",min);
   return 0;
```

}
OUTPUT:



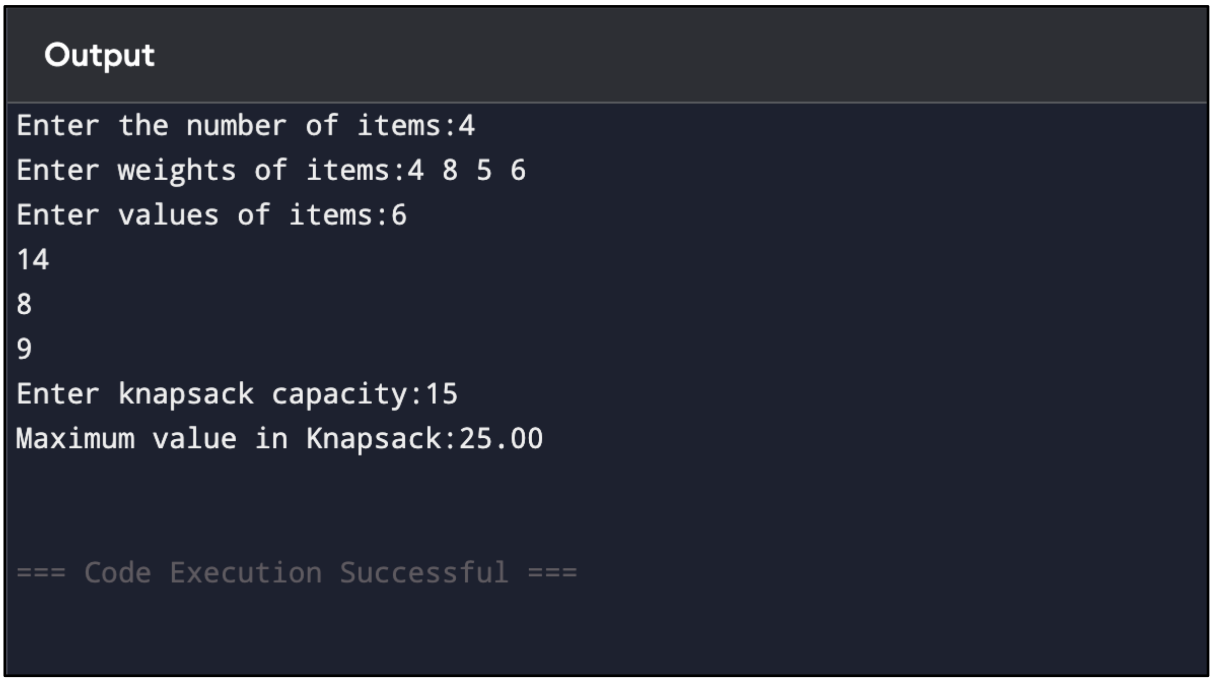
```
Output
Enter the number of elements:5
Enter the elements:7 2 90 21 1
Maximum value:90
Minimum value:1


=== Code Execution Successful ===
```

19.Knapsack using Greedy method

CODE:

```
#include<stdio.h>
int main(){
   int n;
   printf("Enter the number of items:");
   scanf("%d",&n);
   int weight[n],value[n],i,j;
   float ratio[n],temp;
   printf("Enter weights of items:");
   for(i=0;i<n;i++)scanf("%d",&weight[i]);
   printf("Enter values of items:");
   for(i=0;i<n;i++)scanf("%d",&value[i]);
   for(i=0;i<n;i++)ratio[i]=(float)value[i]/weight[i];
   for(i=0;i<n-1;i++){
      for(j=0;j<n-i-1;j++){
         if(ratio[j]<ratio[j+1]){
            temp=ratio[j],ratio[j]=ratio[j+1],ratio[j+1]=temp;
            temp=weight[j],weight[j]=weight[j+1],weight[j+1]=temp;
            temp=value[j],value[j]=value[j+1],value[j+1]=temp;
         }
      }
   }
   int capacity;
   printf("Enter knapsack capacity:");
   scanf("%d",&capacity);
   float totalValue=0.0;
   for(i=0;i<n&&capacity>0;i++){
      if(weight[i]<=capacity){
            totalValue+=value[i];
            capacity-=weight[i];
        }else{
            totalValue+=capacity*ratio[i];
            capacity=0;
        }
    }
    printf("Maximum value in Knapsack:%.2f\n",totalValue);
    return 0;
}
```

OUTPUT:



Output

```
Enter the number of items:4
Enter weights of items:4 8 5 6
Enter values of items:6
14
8
9
Enter knapsack capacity:15
Maximum value in Knapsack:25.00


=== Code Execution Successful ===
```

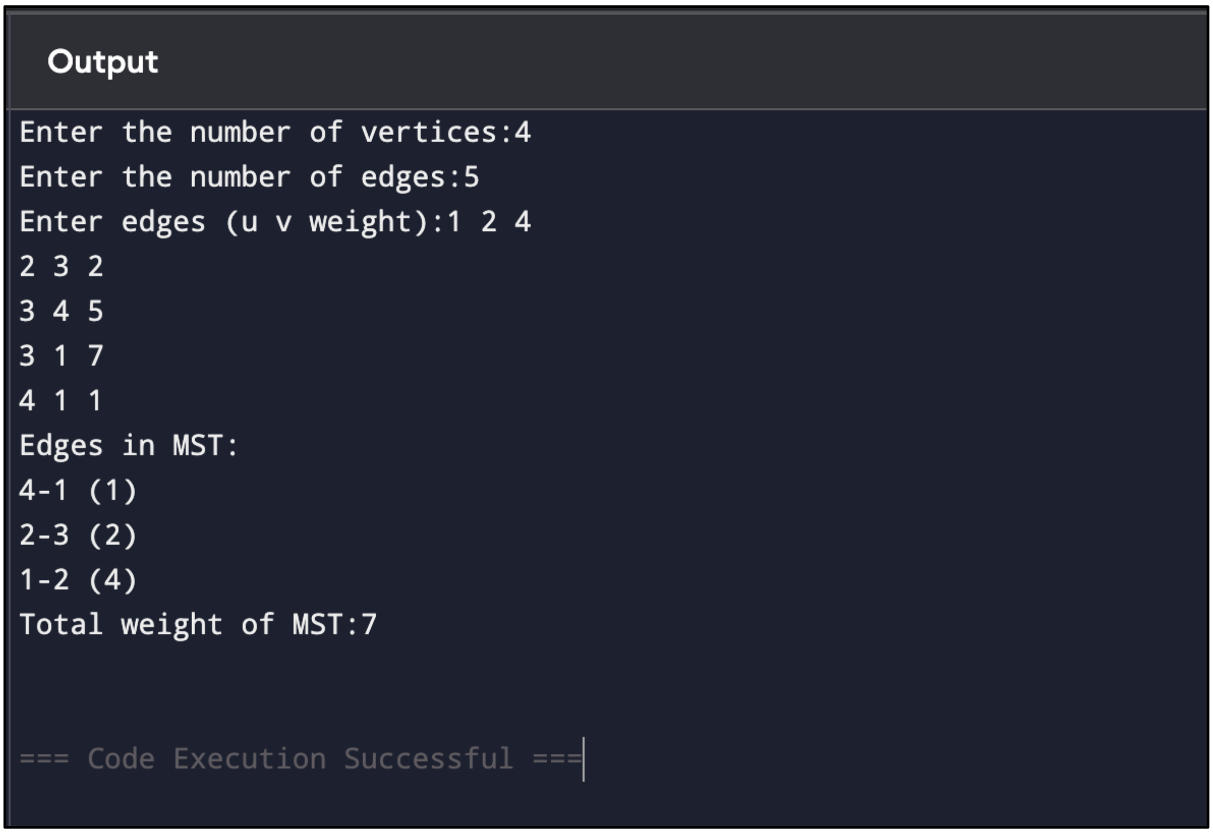
20.MST using Greedy Technique

CODE:

```
#include<stdio.h>
int main(){
   int n,e;
   printf("Enter the number of vertices:");
   scanf("%d",&n);
   printf("Enter the number of edges:");
   scanf("%d",&e);
   int edges[e][3],i,j,parent[n],u,v;
   printf("Enter edges (u v weight):");
   for(i=0;i<e;i++)scanf("%d%d%d",&edges[i][0],&edges[i][1],&edges[i][2]);
   for(i=0;i<e-1;i++){
      for(j=0;j<e-i-1;j++){
         if(edges[j][2]>edges[j+1][2]){
            int temp0=edges[j][0],temp1=edges[j][1],temp2=edges[j][2];
            edges[j][0]=edges[j+1][0];
            edges[j][1]=edges[j+1][1];
            edges[j][2]=edges[j+1][2];
            edges[j+1][0]=temp0;
            edges[j+1][1]=temp1;
            edges[j+1][2]=temp2;
         }
      }
   }
   for(i=0;i<n;i++)parent[i]=i;
   int find(int v){
      while(parent[v]!=v)v=parent[v];
```

```
        return v;
    }
    void unionSets(int u,int v){
        parent[find(u)]=find(v);
    }
    int mstWeight=0;
    printf("Edges in MST:\n");
    for(i=0;i<e;i++){
        u=edges[i][0];
        v=edges[i][1];
        if(find(u)!=find(v)){
            printf("%d-%d (%d)\n",u,v,edges[i][2]);
            mstWeight+=edges[i][2];
            unionSets(u,v);
        }
    }
    printf("Total weight of MST:%d\n",mstWeight);
    return 0;
}
```

OUTPUT:



Output

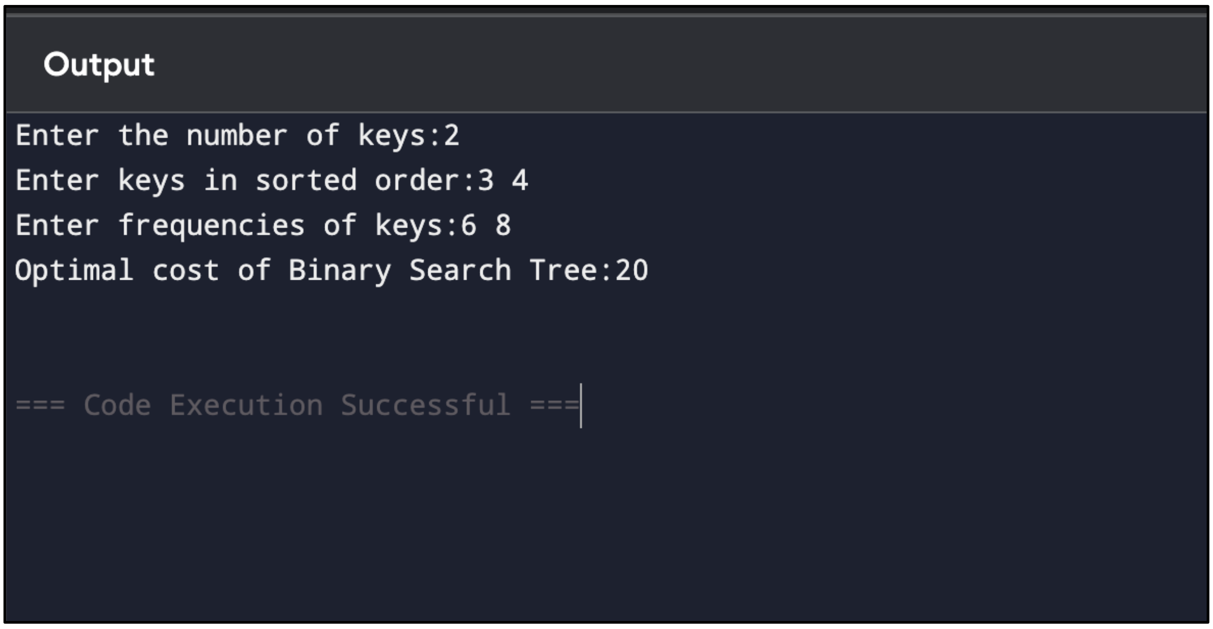
```
Enter the number of vertices:4
Enter the number of edges:5
Enter edges (u v weight):1 2 4
2 3 2
3 4 5
3 1 7
4 1 1
Edges in MST:
4-1 (1)
2-3 (2)
1-2 (4)
Total weight of MST:7


=== Code Execution Successful ===
```

21.Optimal Binary Search Tree using Dynamic Programming

CODE:

```
#include<stdio.h>
int main(){
    int n;
    printf("Enter the number of keys:");
    scanf("%d",&n);
    int keys[n+1],freq[n+1];
    printf("Enter keys in sorted order:");
    for(int i=1;i<=n;i++)scanf("%d",&keys[i]);
    printf("Enter frequencies of keys:");
    for(int i=1;i<=n;i++)scanf("%d",&freq[i]);

    int cost[n+1][n+1],sum[n+1][n+1];
    for(int i=1;i<=n;i++){
        cost[i][i]=freq[i];
        sum[i][i]=freq[i];
    }

    for(int l=2;l<=n;l++){
        for(int i=1;i<=n-l+1;i++){
            int j=i+l-1;
            sum[i][j]=sum[i][j-1]+freq[j];
            cost[i][j]=999999; // Representing infinity
            for(int r=i;r<=j;r++){
                int c=(r>i?cost[i][r-1]:0)+(r<j?cost[r+1][j]:0)+sum[i][j];
                if(c<cost[i][j])cost[i][j]=c;
            }
        }
    }

    printf("Optimal cost of Binary Search Tree:%d\n",cost[1][n]);
    return 0;
}
```

OUTPUT:



Output

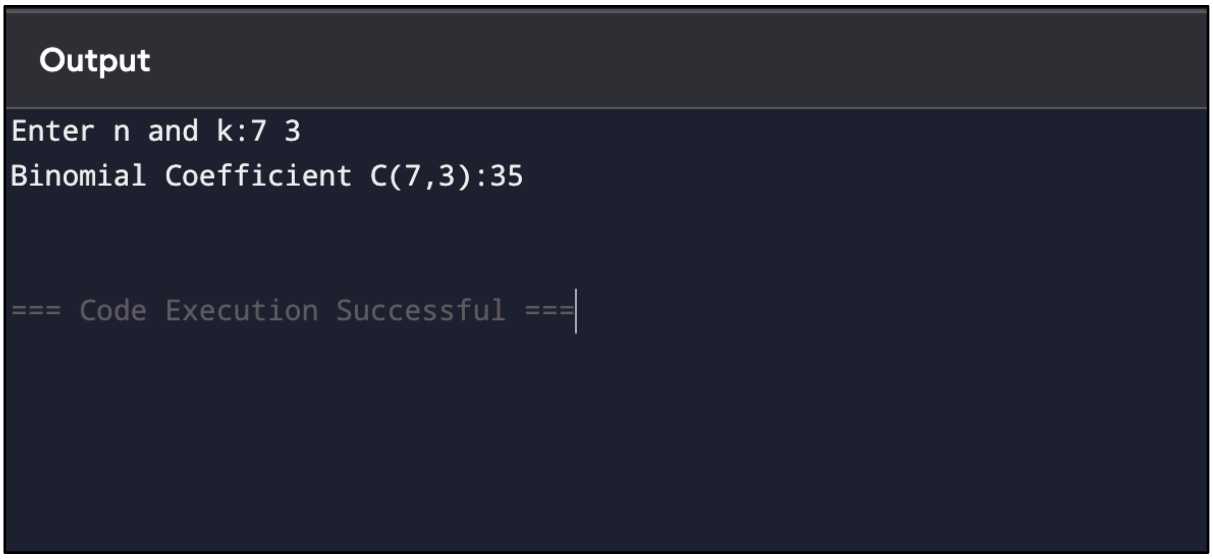
```
Enter the number of keys:2
Enter keys in sorted order:3 4
Enter frequencies of keys:6 8
Optimal cost of Binary Search Tree:20


=== Code Execution Successful ===
```

22.Binary Coefficient using Dynamic Programming

CODE:

```
#include<stdio.h>
int main(){
   int n,k;
   printf("Enter n and k:");
   scanf("%d%d",&n,&k);
   int C[n+1][k+1];
   for(int i=0;i<=n;i++){
      for(int j=0;j<=k;j++){
         if(j==0||j==i)C[i][j]=1;
         else C[i][j]=C[i-1][j-1]+C[i-1][j];
      }
   }
   printf("Binomial Coefficient C(%d,%d):%d\n",n,k,C[n][k]);
   return 0;
}
```

OUTPUT:


Output

```
Enter n and k:7 3
Binomial Coefficient C(7,3):35


=== Code Execution Successful ===
```

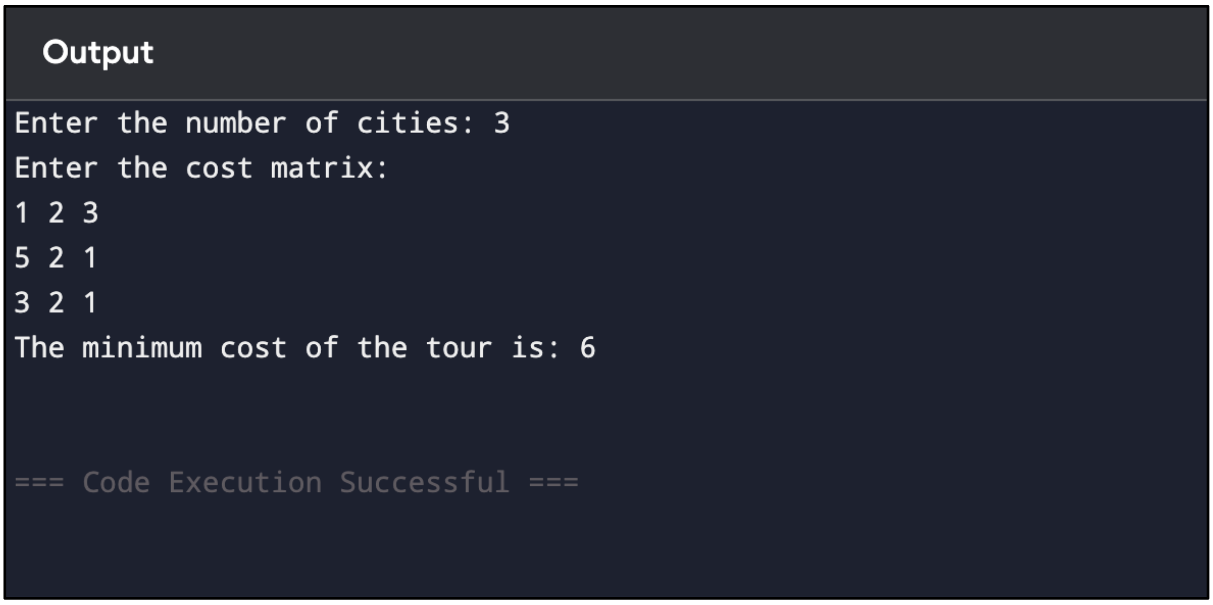
25.TSP using Dynamic Programming

CODE:

```
#include<stdio.h>
#define INF 999999

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

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

```
    int dp[1<<n][n];
    for(int mask=0;mask<(1<<n);mask++){
        for(int i=0;i<n;i++)dp[mask][i]=INF;
    }

    dp[1][0]=0; // Starting city is 0

    for(int mask=1;mask<(1<<n);mask++){
        for(int i=0;i<n;i++){
            if(!(mask&(1<<i)))continue;
            for(int j=0;j<n;j++){
                if(mask&(1<<j))continue;
dp[mask|(1<<j)][j]=dp[mask|(1<<j)][j]<dp[mask][i]+cost[i][j]?dp[mask|(1<<j)
][j]:dp[mask][i]+cost[i][j];
            }
        }
    }

    int res=INF;
    for(int        i=1;i<n;i++)res=res<dp[(1<<n)-1][i]+cost[i][0]?res:dp[(1<<n)-
1][i]+cost[i][0];

    printf("The minimum cost of the tour is: %d\n",res);
    return 0;
}
```

OUTPUT:

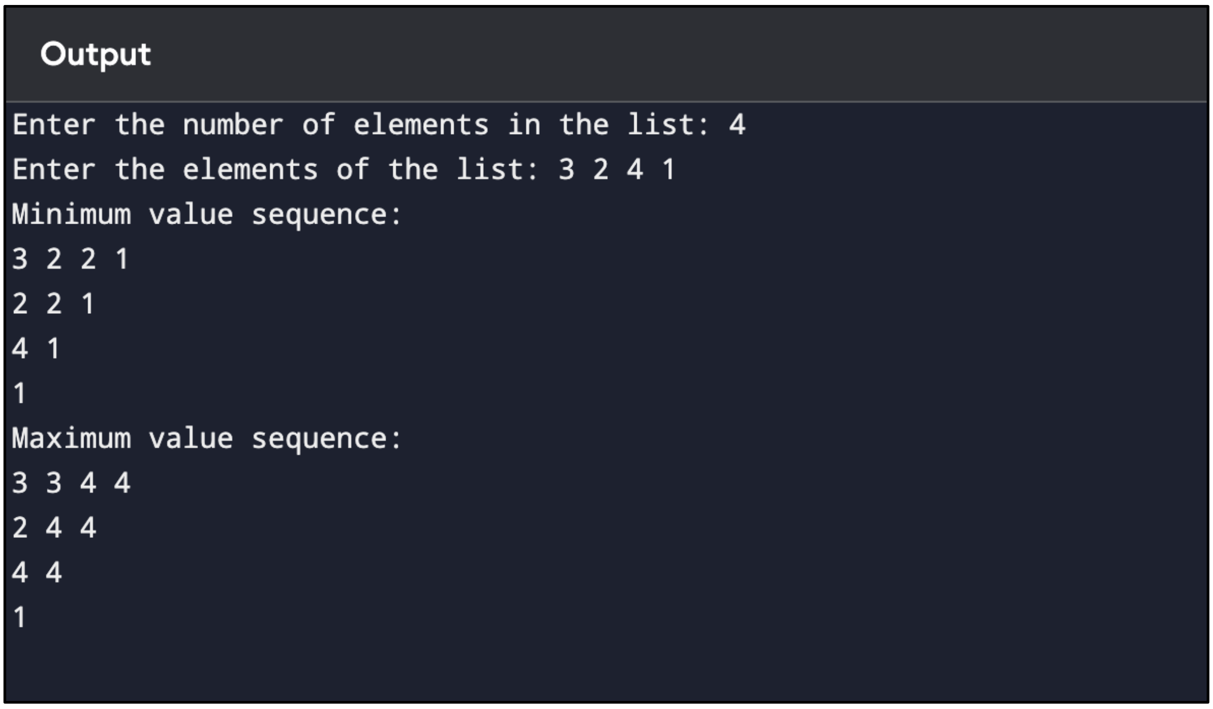

Output
```
Enter the number of cities: 3
Enter the cost matrix:
1 2 3
5 2 1
3 2 1
The minimum cost of the tour is: 6


=== Code Execution Successful ===
```

31.MIN and Max Sequence for all numbers list

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

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

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

    printf("Minimum value sequence:\n");
    for(int i = 0; i < n; i++) {
        int min = arr[i];
        for(int j = i; j < n; j++) {
            if(arr[j] < min) {
                min = arr[j];
            }
            printf("%d ", min);
        }
        printf("\n");
    }

    printf("Maximum value sequence:\n");
    for(int i = 0; i < n; i++) {
        int max = arr[i];
        for(int j = i; j < n; j++) {
            if(arr[j] > max) {
                max = arr[j];
            }
            printf("%d ", max);
        }
        printf("\n");
    }

    return 0;
}
```

OUTPUT:


Output

```
Enter the number of elements in the list: 4
Enter the elements of the list: 3 2 4 1
Minimum value sequence:
3 2 2 1
2 2 1
4 1
1
Maximum value sequence:
3 3 4 4
2 4 4
4 4
1
```

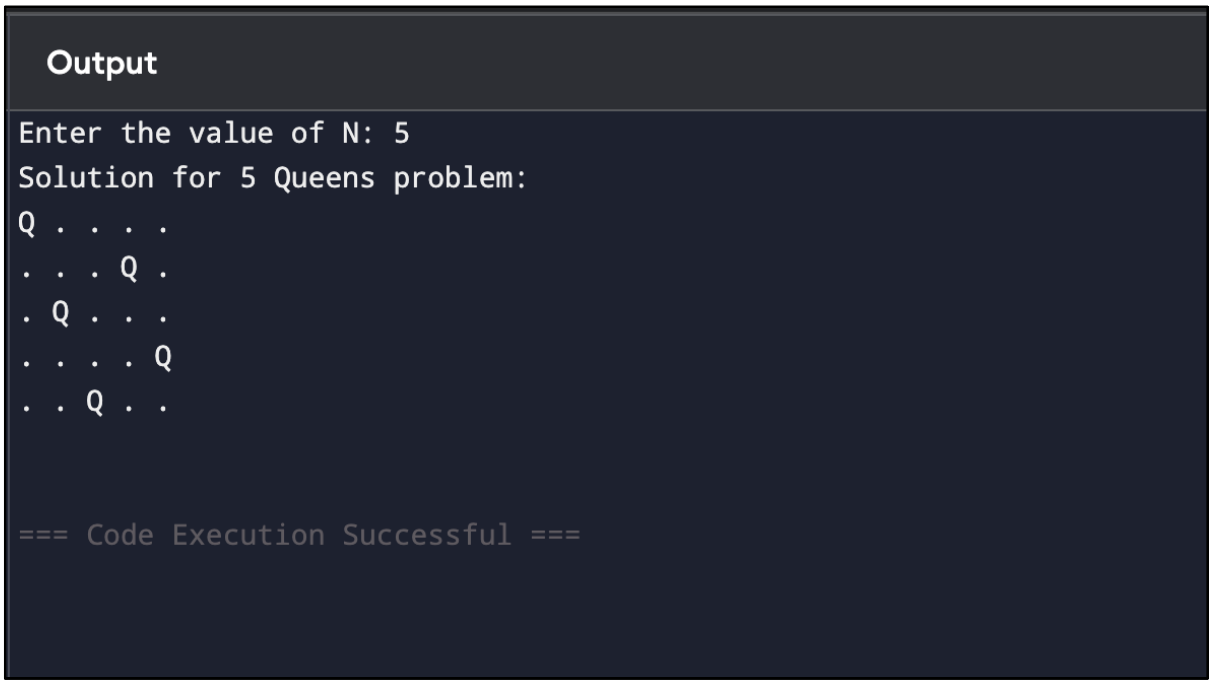
32.N Queens using Backtracking technique

CODE:

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

int board[MAX][MAX];

int isSafe(int row,int col,int n){
   for(int i=0;i<col;i++){
      if(board[row][i])return 0;
   }
   for(int i=row,j=col;i>=0&&j>=0;i--,j--){
      if(board[i][j])return 0;
   }
   for(int i=row,j=col;i<n&&j>=0;i++,j--){
      if(board[i][j])return 0;
   }
   return 1;
}

int solveNQueens(int col,int n){
   if(col>=n)return 1;
   for(int i=0;i<n;i++){
      if(isSafe(i,col,n)){
         board[i][col]=1;
         if(solveNQueens(col+1,n))return 1;
         board[i][col]=0;
      }
   }
   return 0;
}

int main(){
   int n;
   printf("Enter the value of N: ");
   scanf("%d",&n);
   for(int i=0;i<n;i++){
      for(int j=0;j<n;j++)board[i][j]=0;
   }
```

```
    if(solveNQueens(0,n)){
        printf("Solution for %d Queens problem:\n",n);
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(board[i][j])printf("Q ");
                else printf(". ");
            }
            printf("\n");
        }
    }else{
        printf("No solution exists for %d Queens problem.\n",n);
    }
    return 0;
}
```

OUTPUT:



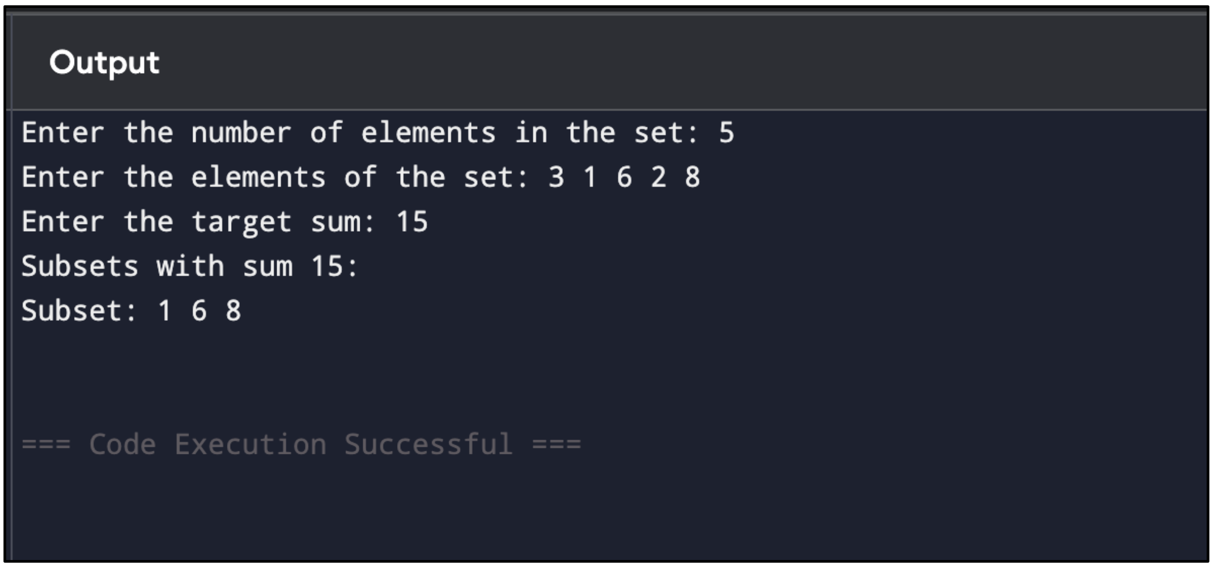
```
Enter the value of N: 5
Solution for 5 Queens problem:
Q . . . .
. . . Q .
. Q . . .
. . . . Q
. . Q . .


=== Code Execution Successful ===
```

34.Sum of Subsets using Backtracking Technique

CODE:

```
#include<stdio.h>
int n, target, found = 0;
void sumOfSubsets(int arr[], int subset[], int idx, int curr_sum, int start) {
    if(curr_sum == target) {
        found = 1;
        printf("Subset: ");
        for(int i = 0; i < idx; i++) {
            printf("%d ", subset[i]);
        }
        printf("\n");
        return;
    }
    for(int i = start; i < n; i++) {
        if(curr_sum + arr[i] <= target) {
            subset[idx] = arr[i];
            sumOfSubsets(arr, subset, idx + 1, curr_sum + arr[i], i + 1);
        }
    }
}
int main() {
    printf("Enter the number of elements in the set: ");
    scanf("%d", &n);

    int arr[n], subset[n];
    printf("Enter the elements of the set: ");
    for(int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter the target sum: ");
    scanf("%d", &target);
    printf("Subsets with sum %d:\n", target);
    sumOfSubsets(arr, subset, 0, 0, 0);
    if(!found) {
        printf("No subsets found with the given sum.\n");
    }
     return 0;
}
```

OUTPUT:



```
Enter the number of elements in the set: 5
Enter the elements of the set: 3 1 6 2 8
Enter the target sum: 15
Subsets with sum 15:
Subset: 1 6 8


=== Code Execution Successful ===
```

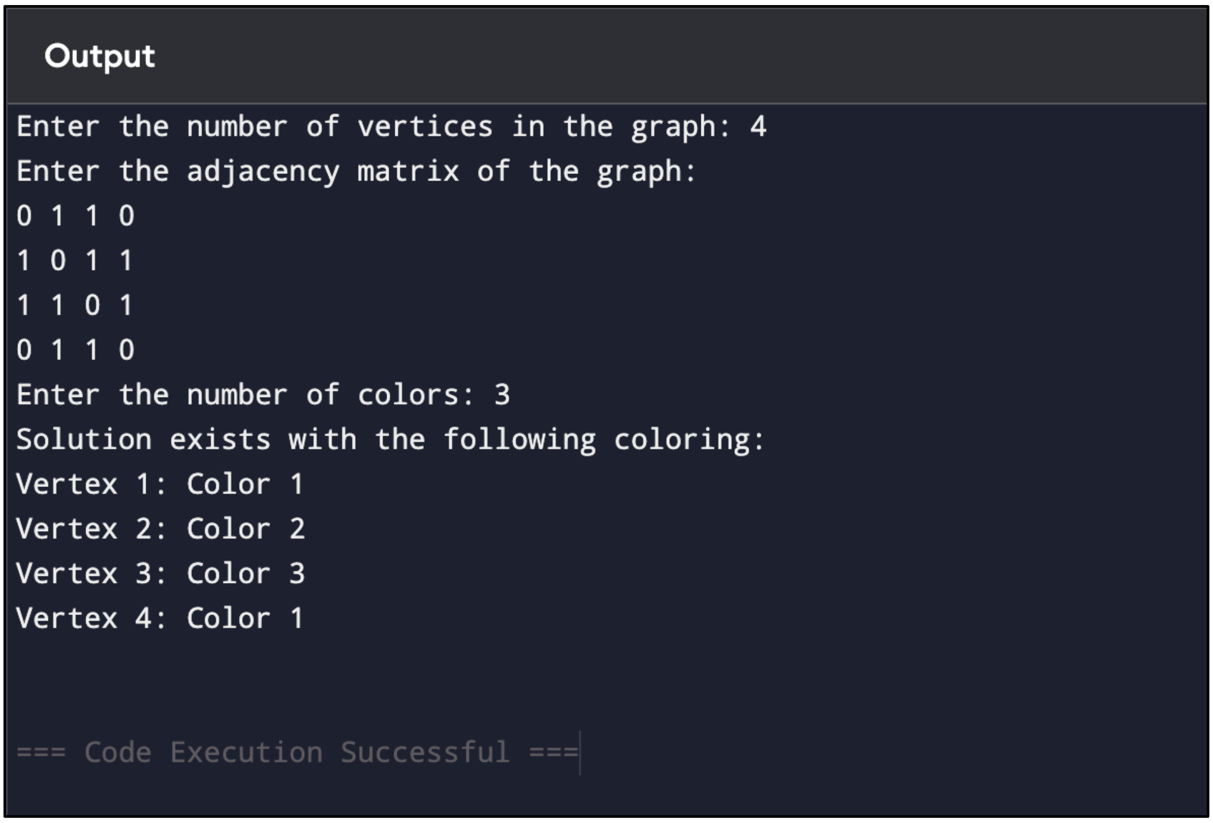
35.Graph Coloring using Backtracking

CODE:

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

int n, graph[MAX][MAX], colors[MAX], m;

int isSafe(int node, int c) {
   for (int i = 0; i < n; i++) {
      if (graph[node][i] && colors[i] == c) return 0;
   }
   return 1;
}

int graphColoring(int node) {
   if (node == n) return 1;
   for (int c = 1; c <= m; c++) {
      if (isSafe(node, c)) {
         colors[node] = c;
         if (graphColoring(node + 1)) return 1;
         colors[node] = 0;
      }
   }
   return 0;
}

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

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

```
    printf("Enter the number of colors: ");
    scanf("%d", &m);

    for (int i = 0; i < n; i++) colors[i] = 0;

    if (graphColoring(0)) {
        printf("Solution exists with the following coloring:\n");
        for (int i = 0; i < n; i++) {
            printf("Vertex %d: Color %d\n", i + 1, colors[i]);
        }
    } else {
        printf("No solution exists with the given number of colors.\n");
    }
    return 0;
}
```

OUTPUT:



```
Enter the number of vertices in the graph: 4
Enter the adjacency matrix of the graph:
0 1 1 0
1 0 1 1
1 1 0 1
0 1 1 0
Enter the number of colors: 3
Solution exists with the following coloring:
Vertex 1: Color 1
Vertex 2: Color 2
Vertex 3: Color 3
Vertex 4: Color 1


=== Code Execution Successful ===
```

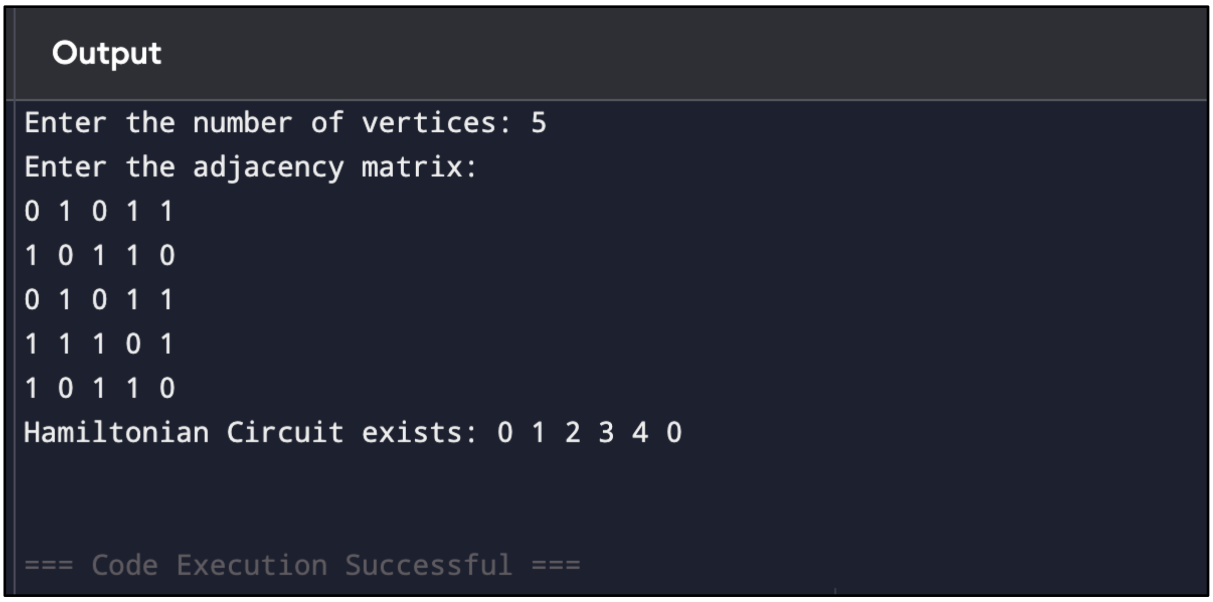
40.Hamiltonian Circuit using Backtracking

CODE:

```
#include<stdio.h>
#define MAX 10
int n,graph[MAX][MAX],path[MAX];
int main(){
   int isSafe(int v,int pos){
      if(graph[path[pos-1]][v]==0)return 0;
      for(int i=0;i<pos;i++){
         if(path[i]==v)return 0;
      }
      return 1;
   }
   int hamiltonian(int pos){
      if(pos==n)return graph[path[pos-1]][path[0]]==1;
      for(int v=1;v<n;v++){
         if(isSafe(v,pos)){
            path[pos]=v;
            if(hamiltonian(pos+1))return 1;
            path[pos]=-1;
         }
      }
      return 0;
   }
   printf("Enter the number of vertices: ");
   scanf("%d",&n);
   printf("Enter the adjacency matrix:\n");
   for(int i=0;i<n;i++){
      for(int j=0;j<n;j++){
         scanf("%d",&graph[i][j]);
      }
   }
   for(int i=0;i<n;i++)path[i]=-1;
```

```
    path[0]=0;
    if(hamiltonian(1)){
        printf("Hamiltonian Circuit exists: ");
        for(int i=0;i<n;i++)printf("%d ",path[i]);
        printf("%d\n",path[0]);
    }
    else{
        printf("No Hamiltonian Circuit exists.\n");
    }
    return 0;
}
```

OUTPUT:



Output

```
Enter the number of vertices: 5
Enter the adjacency matrix:
0 1 0 1 1
1 0 1 1 0
0 1 0 1 1
1 1 1 0 1
1 0 1 1 0
Hamiltonian Circuit exists: 0 1 2 3 4 0


=== Code Execution Successful ===
```