S.No: 1	Exp. Name: ***Program for Recursive Linear search***	Date: 2022-03-11

Aim:

Program for Recursive liner research

Source Code:

linearSearch.c

```
#include<stdio.h>
int recursive(int a[],int item, int p, int r){
   if (r<p){
      return 1;
   }
   else if(a[p] == item){
      return p;
   }
   else{
      recursive(a,item,p+1,r);
   }
}
int main(){
   int val,i,item,p,r,n,m;
   printf("enter the no of elements: ");
   scanf("%d",&n);
   int a[n];
   printf("Enter %d integer(s)\n",n);
   for(i =0;i<n;i++){
      scanf("%d",&a[i]);
   }
   printf("enter the item to be search: ");
   scanf("%d",&m);
   p = 0;
   r = n;
   val = recursive(a,m,p,r);
   if (val==1){
      printf("no item found");
   }
   else{
      printf("item location = %d  item = %d",val+1,m);
   }
   return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
enter the no of elements: 5
Enter 5 integer(s) 1

Test Case - 1	
2	2
3	3
6	6
5	5
enter the item to be search: 5	
item location = 5 item = 5	

Test Case - 2	
User Output	
enter the no of elements: 3	
Enter 3 integer(s) 22	
33	33
9	9
enter the item to be search: 0	
no item found	

S.No: 2	Exp. Name: ***Program for Recursive Binary Search***	Date: 2022-04-01

Aim:

Program for Recursive Binary search

Source Code:

binaryrSearch.c

```
#include<stdio.h>
int binary(int arr[],int k,int low,int high){
   if(low<=high){
      int mid = (low+high)/2;
      if(arr[mid]==k){
         return mid;
      }
      else if (arr[mid] < k){
         return binary(arr,k,mid+1,high);
      }
      else{
         return binary(arr,k,low,mid-1);
      }
   }
   return -1;
}

int main(){
   int n,key;
   printf("Enter number of elements: ");
   scanf("%d",&n);
   int arr[n];
   printf("Enter the sorted array: ");
   for(int i=0;i<n;i++){
      scanf("%d",&arr[i]);
   }
   printf("enter the item to be search: ");
   scanf("%d",&key);

   int bin = binary(arr,key,0,n-1);
   if (bin == -1){
      printf("item not present");
   }
   else{
      printf("item present");
   }
   return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

Test Case - 1
Enter number of elements: 5
Enter the sorted array: 1 12 22 32 45
enter the item to be search: 12
item present

Test Case - 2
User Output
Enter number of elements: 2
Enter the sorted array: 0 12
enter the item to be search: 1
item not present

S.No: 3	Exp. Name: ***program to sort a list of elements using insertion sort***	Date: 2022-04-01

Aim:

Program to sort a list of elements using Insertion sort.

Source Code:

InsertionSort.c

```
#include<stdio.h>
int insertion(int arr[],int n)
{
   for(int i = 1;i<n;i++){
      int key = arr[i];
      int j = i-1;
      while((j>=0) && (arr[j]>key) ){
         arr[j+1] = arr[j];
         j--;
      }
      arr[j+1] = key;
   }
}

int main(){
   int size;
   printf("Enter size of the array: ");
   scanf("%d",&size);
   int arr[size];
   printf("Enter %d elements in to the array: ",size);
   for(int i = 0;i<size;i++){
      scanf("%d",&arr[i]);
   }

   insertion(arr,size);
   printf("After sorting the elements are:");
   for(int i=0;i<size;i++){
      printf(" %d",arr[i]);
   }
   return 0;
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter size of the array: 5
Enter 5 elements in to the array: 87 12 45 65 21
After sorting the elements are: 12 21 45 65 87

Test Case - 2
User Output

Test Case - 2
Enter size of the array: 3
Enter 3 elements in to the array: 55 14 78
After sorting the elements are: 14 55 78

S.No: 4	Exp. Name: ***Program to sort a list of elements using selection sort***	Date: 2022-04-01

Aim:

Program to sort a list of elements using Selection sort.

Source Code:

selection.c

```
#include<stdio.h>
int selection(int arr[],int n){
   int temp;
   for(int i = 0;i<n;i++){
      for(int j = i+1;j<n;j++){
         if (arr[i] > arr[j]){
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;

         }
      }
   }
}

int main(){
   int size;
   printf("Enter size of the array : ");
   scanf("%d",&size);
   int arr[size];
   printf("Enter the elements :");
   for(int i=0;i<size;i++){
      scanf("%d",&arr[i]);
   }
   selection(arr,size);
   printf("The sorted elements are : ");
   for(int i=0;i<size;i++){
      printf("%d\t",arr[i]);
   }
   return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1	
User Output	
Enter size of the array : 3	
Enter the elements : 6	
2	2
4	4
The sorted elements are : 2 4 6	

Test Case - 2	
User Output	
Enter size of the array : 4	
Enter the elements : 90	
87	87
95	95
92	92
The sorted elements are : 87 90 92 95	

S.No: 5	Exp. Name: ***Program to implementation of counting sort***	Date: 2022-04-08

ID: 2001330100206

Aim:

Program to sort a list of elements using Counting sort.

Source Code:

countSort.c

```
#include<stdio.h>
void countsort(int arr[],int n){
   int k=arr[0];
   for(int i = 0;i<n;i++)
      k = k<arr[i]?arr[i]:k;
   int count[123] = {0};
   for(int i = 0;i<n;i++)
      count[arr[i]]++;
   for(int i = 1;i<=k;i++)
      count[i]+=count[i-1];
   int output[n];
   for(int i = n-1;i>=0;i--)
      output[--count[arr[i]]] = arr[i];
   for(int i =0;i<n;i++)
      arr[i] = output[i];
}

int main(){
   int n;
   printf("enter the no. of arry element: ");
   scanf("%d",&n);
   int arr[n];
   printf("enter the element: ");
   for(int i =0;i<n;i++)
      scanf("%d",&arr[i]);
   countsort(arr,n);
   for(int i = 0;i<n;i++)
      printf("%d ",arr[i]);
      return 0;
}
```

2020-2024-CSE-C2

Noida Institute of Engineering and Technology

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
enter the no. of arry element: 3
enter the element: 12 3 65
3 12 65

S.No: 6	Exp. Name: ***Program to sort a list of elements using Merge Sort***	Date: 2022-05-03

Aim:

Program to sort a list of elements using Merge Sort

Source Code:

Merge.c

```
#include<stdio.h>
void mergesort(int a[],int i,int j);
void merge(int a[],int i1,int j1,int i2,int j2);
int main() {
   int a[30],n,i;
   printf(" Enter How many Numbers : ");
   scanf("%d",&n);
   printf(" Enter %d Numbers :",n);
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   mergesort(a,0,n-1);
   printf(" Sorted Numbers are : ");
   for(i=0;i<n;i++)
   printf("%d\t",a[i]);
   return 0;
}
void mergesort(int a[],int i,int j) {
   int mid;
   if(i<j)
   {
      mid=(i+j)/2;
      mergesort(a,i,mid);
      mergesort(a,mid+1,j);
      merge(a,i,mid,mid+1,j);
   }
}
void merge(int a[],int i1,int j1,int i2,int j2)
{
   int temp[50];
   int i,j,k;
   i=i1;
   j=i2;
   k=0;
   while(i<=j1 && j<=j2) {
      if(a[i]<a[j])
      temp[k++]=a[i++];
      else
      temp[k++]=a[j++];
   }
   while(i<=j1)
   temp[k++]=a[i++];
   while(j<=j2)
   temp[k++]=a[j++];
   for(i=i1,j=0;i<=j2;i++,j++)
   a[i]=temp[j];
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter How many Numbers : 6
Enter 6 Numbers : 12 10 5 4 3 1
Sorted Numbers are : 1 3 4 5 10 12

Test Case - 2
User Output
Enter How many Numbers : 4
Enter 4 Numbers : -8 -4 1 2
Sorted Numbers are : -8 -4 1 2

S.No: 7	Exp. Name: ***Program to sort a list of elements using Quick Sort***	Date: 2022-05-03

ID: 2001330100206

Aim:

Program to sort a list of elements using Quick Sort

Source Code:

QuickSort.c

```
#include <stdio.h>
void quicksort (int [], int, int);
int main() {
   int list[50];
   int size, i;
   printf("Enter Number of elements : ");
   scanf("%d", &size);
   printf("Enter %d Elements : ",size);
   for (i = 0; i < size; i++) {
      scanf("%d", &list[i]);
   }
   quicksort(list, 0, size - 1);
   printf("Sorted Numbers are : ");
   for (i = 0; i < size; i++) {
      printf("%d ", list[i]);
   }
   printf("\n");
   return 0;

}
void quicksort(int list[], int low, int high) {
   int pivot, i, j, temp;
   if (low < high) {
      pivot = low;
      i = low;
      j = high;
      while (i < j) {
         while (list[i] <= list[pivot] && i <= high) {
            i++;
         }
         while (list[j] > list[pivot] && j >= low) {
            j--;
         }
         if (i < j) {
            temp = list[i];
            list[i] = list[j];
            list[j] = temp;
         }
      }

      temp = list[j];
      list[j] = list[pivot];
      list[pivot] = temp;
      quicksort(list, low,j - 1);
      quicksort(list, j + 1, high);
```

2020-2024-CSE-C2

Noida Institute of Engineering and Technology

```
    }
    }
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter Number of elements : 5
Enter 5 Elements : 37 2 4 5 1
Sorted Numbers are : 1 2 4 5 37

Test Case - 2
User Output
Enter Number of elements : 9
Enter 9 Elements : 59 87 36 1 2 54 48 9 18
Sorted Numbers are : 1 2 9 18 36 48 54 59 87

S.No: 8	Exp. Name: ***Program to sort a list of elements using Heap Sort***	Date: 2022-05-03

ID: 2001330100206

Aim:

Program to sort a list of elements using Heap Sort

Source Code:

heapSort.c

```
#include<stdio.h>
void main() {
   int heap[10], no, i, j, c, root, temp;
   printf("enter the no. of element: ");
   scanf("%d", &no);
   printf("Enter elements: ");
   for(i=0; i<no; i++)
   scanf("%d", &heap[i]);
   for(i=1; i<no; i++) {
      c=i;
      do{
         root=(c-1)/2;
         if(heap[root]<heap[c]) {
            temp=heap[root];
            heap[root]=heap[c];
            heap[c]=temp;
         }
         c=root;
      }while (c!=0); }
      for (j=no-1; j>=0;j--) {
         temp=heap[0];
         heap[0]=heap[j];
         heap[j]=temp;
         root=0;
         do{
            c=2*root+1;
            if((heap[c]<heap[c+1])&&c<j-1)
            c++;
            if(heap[root]<heap[c]&&c<j) {
               temp=heap[root];
               heap[root]=heap[c];
               heap[c]=temp;
            }
            root=c; }
            while(c<j);
      }
      for(i=0;i<no; i++)
      printf("%d\t", heap[i]);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

Test Case - 1
User Output
enter the no. of element: 3
Enter elements: 45 6 12
6 12 45

S.No: 9	Exp. Name: ***Maximun and minimum element using divide and conquor***	Date: 2022-05-03

ID: 2001330100206

Aim:

Program to compute Maximum and Minimum element using divide and conquer

Source Code:

divideAndConquer.c

```
#include<stdio.h>
#include<conio.h>
void main(){
   int n;
   printf("Enter the total number of Elements : ");
   scanf("%d",&n);
   printf("Enter the numbers : ");
   int a[n];
   for(int i=0;i<n;i++){
      scanf("%d",&a[i]);
   }
   int min=a[0];
   int max=a[0];
   for(int i=0;i<n;i++){
      if(a[i]<min){
         min=a[i];
      }
      if(a[i]>max){
         max=a[i];
      }
   }
   printf("Minimum element in an array : %d\n",min);
   printf("Maximum element in an array : %d\n",max);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the total number of Elements : 6
Enter the numbers : 6 7 23 1 89 45
Minimum element in an array : 1
Maximum element in an array : 89

Test Case - 2
User Output
Enter the total number of Elements : 10
Enter the numbers : 10 12 65 87 98 56 32 54 43 21
Minimum element in an array : 10
Maximum element in an array : 98

2020-2024-CSE-C2

Noida Institute of Engineering and Technology

S.No: 10	Exp. Name: ***Program to compute Optimal Paranthesization for given Matrix chain order***	Date: 2022-05-03

Aim:

Program to compute Optimal Paranthesization for given Matrix chain order

Source Code:

chainMultiplication.c

```
#include<stdio.h>
#include<limits.h>
int i,L,j,k,q;
int m[20][20],S[20][20];
void optimal(int,int);
void matrix_chain(int p[],int n){
   for(i=1;i<=n;i++){
      m[i][i]=0;
   }
   for(L=2;L<=n;L++){
      for(i=1;i<=n-L+1;i++){
         j=i+L-1;
         m[i][j]=INT_MAX;
         for(k=i;k<=j-1;k++){
            q=m[i][k]+m[k+1][j]+(p[i-1]*p[k]*p[j]);
            if(q<m[i][j]){
               m[i][j]=q;
               S[i][j]=k;
            }
         }
      }
   }
   optimal(1,n);
}
void optimal(int i, int j){
   if(i==j){
      printf("A%d",i);
   }
   else{
      printf("(");
      optimal(i,S[i][j]);
      optimal(S[i][j]+1,j);
      printf(")");
   }
}
void main(){
   int n;
   printf("enter the matrices");
   scanf("%d",&n);
   int p[10];
   for(int i=0;i<=n;i++){
      scanf("%d",&p[i]);
   }
   matrix_chain(p,n);
   printf("%d",m[1][n]);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1	
User Output	
enter the matrices 3	
4	4
5	5
6	6
7	7
((A1A2)A3)288	

S.No: 11	Exp. Name: ***Program to compute Longest Common Subsequence of two given Sequences***	Date: 2022-05-03

Aim:

Program to compute Longest Common Subsequence of two given Sequences

Source Code:

largestSubSequence.c

```
#include<stdio.h>
#include<conio.h>
void lcs(char a[], char b[])
{
   int n= strlen(a);
   int m=strlen(b);
   int c[n+1][m+1];
   for(int j=0;j<=m;j++){
      c[0][j]=0;
   }
   for(int i=1;i<=n;i++)
   {
      c[i][0]=0;
   }
   for(int i=1;i<=n;i++)
   {
      for(int j=1;j<=m;j++)
      {
         if(a[i-1]==b[j-1])
         {
            c[i][j]=c[i-1][j-1]+1;
         }
         else if (c[i-1][j]>=c[i][j-1])
         {
            c[i][j]=c[i-1][j];
         }
         else
         {
            c[i][j]=c[i][j-1];
         }
      }
   }
   printf("Length of LCS is %d\n",c[n][m]);
}
void main()
{
   char a[50],b[50];
   printf("Enter a string1: ");
   gets(a);
   printf("Enter a string2: ");
   gets(b);
   lcs(a,b);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a string1: adgfh
Enter a string2: sf
Length of LCS is 1

Test Case - 2
User Output
Enter a string1: aggtab
Enter a string2: gxtxayab
Length of LCS is 4

S.No: 12	Exp. Name: ***Program to implement, 0/1 Knapsack problem using Dynamic Programming***	Date: 2022-05-03

Aim:

Write a program to implement, 0/1 Knapsack problem using Dynamic Programming

Source Code:

Knapsack.c

```
#include<stdio.h>
int max(int a,int b){
   return(a>b)?a:b;
}
int knapsack(int W,int v[],int w[],int n){
   if(n==0|W==0)
   return 0;
   if(w[n-1]>W)
   return knapsack(W,v,w,n-1);
   else
   return max(v[n-1]+knapsack(W-w[n-1],v,w,n-1),knapsack(W,v,w,n-1));
}
void main(){
   int n,W;
   printf("Enter number of items:");
   scanf("%d",&n);
   int v[n],w[n];
   printf("Enter value and weight of items:");
   for(int i=0;i<n;i++){
      scanf("%d %d",&v[i],&w[i]);
   }
   printf("Enter size of knapsack:");
   scanf("%d",&W);
   printf("Maximum value in 0/1 knapsack :%d",knapsack(W,v,w,n));
}
```

Execution Results - All test cases have succeeded!

Test Case - 1	
User Output	
Enter number of items: 3	
Enter value and weight of items: 100 20	
50 10	50 10
150 30	150 30
Enter size of knapsack: 50	
Maximum value in 0/1 knapsack :250	

Test Case - 2
User Output
Enter number of items: 4

Test Case - 2	
Enter value and weight of items: 10 23	
20 5	20 5
30 6	30 6
40 9	40 9
Enter size of knapsack: 50	
Maximum value in 0/1 knapsack :100	

S.No: 13	Exp. Name: ***Program to find All-Pairs Shortest Paths problem using Floyd's algorithm.***	Date: 2022-05-03

ID: 2001330100206

Aim:

Program to Implement All-Pairs Shortest Paths problem using Floyd's algorithm

Source Code:

Floyds.c

```
#include<stdio.h>
#include<conio.h>
#include<limits.h>
int p[20][20];
int d[20][20];
int w[20][20];
void print_path(int i, int j)
{
   if(i==j)
   printf("%d",i);
   else
   {
      if(p[i][j]==-1)
      printf("No path Exists");
      else
      {
         print_path(i,p[i][j]);
         printf("-> %d",j);
      }
   }
}
void warshall(int n)
{
   for(int i=1;i<=n;i++)
   {
      for(int j=1;j<=n;j++)
      {
         d[i][j]=w[i][j];
      }
   }
   for(int k=1;k<=n;k++)
   {
      for(int i=1;i<=n;i++)
      {
         for(int j=1; j<=n;j++)
         {
            if(d[i][k]==INT_MAX || d[k][j]==INT_MAX)
            continue;
            if(d[i][k]+d[k][j]<d[i][j])
            {
               d[i][j]=d[i][k]+d[k][j];
               p[i][j]=p[k][j];
            }
         }
      }
   }
```

```
}
void main()
{
   int i,j,v,s,des;
   char ch;
   printf("Enter number of vertices: ");
   scanf("%d",&v);
   printf("Enter the weight matrix");
   for(i=1;i<=v;i++)
   {
      for(j=1;j<=v;j++)
      {
         if(i==j)
         {
            w[i][j]=0;
            p[i][j]=-1;
            continue;
         }
         printf("Is edge (%d,%d) present in graph (y/n): ",i,j);
         fflush(stdin);
         scanf("%c",&ch);
         if(ch=='y' || ch=='Y')
         {
            printf("Enter weight of edge (%d,%d): ",i,j);
            scanf("%d",&w[i][j]);
            p[i][j]=i;
         }
         else
         {
            w[i][j]=INT_MAX;
            p[i][j]=-1;
         }
      }
   }
   warshall(v);
   printf("Enter source and destination: ");
   scanf("%d %d",&s,&des);
   printf("Distance = %d",d[s][des]);
   print_path(s,des);
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter number of vertices: 3
Enter the weight matrixIs edge (1,2) present in graph (y/n): y
Enter weight of edge (1,2): 10
Is edge (1,3) present in graph (y/n): 5
Is edge (2,1) present in graph (y/n): n
Is edge (2,3) present in graph (y/n): y
Enter weight of edge (2,3): 10

Test Case - 1
Is edge (3,1) present in graph (y/n): y
Enter weight of edge (3,1): 5
Is edge (3,2) present in graph (y/n): y
Enter weight of edge (3,2): 15
Enter source and destination: 1 3
Distance = 201-> 2-> 3

Test Case - 2
User Output
Enter number of vertices: 2
Enter the weight matrixIs edge (1,2) present in graph (y/n): y
Enter weight of edge (1,2): 5
Is edge (2,1) present in graph (y/n): y
Enter weight of edge (2,1): 20
Enter source and destination: 1 2
Distance = 51-> 2

S.No: 14	Exp. Name: ***Program to implement N-Queen's problem using backtracking***	Date: 2022-05-03

ID: 2001330100206

Aim:

Program to implement N-Queen's problem using backtracking

Source Code:

nQueen.c

```
#include<stdio.h>
#include<math.h>
int board[20],count;
int main()
{
   int n,i,j;
   void queen(int row,int n);
   printf("Enter number of Queens: ");
   scanf("%d",&n);
   queen(1,n);
   return 0;
}
void print(int n)
{
   int i,j;
   for(i=1;i<=n;++i)
   {
      for(j=1;j<=n;++j)
      {
         if(board[i]==j)
         printf("row no %d\tcolom no %d\n",i,j);
      }
   }
}
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)
```

2020-2024-CSE-C2

Noida Institute of Engineering and Technology

```
            print(n);
            else
            queen(row+1,n);
        }
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter number of Queens: 4
row no 1 colom no 2
row no 2 colom no 4
row no 3 colom no 1
row no 4 colom no 3
row no 1 colom no 3
row no 2 colom no 1
row no 3 colom no 4
row no 4 colom no 2

S.No: 15	Exp. Name: ***Program to find the solution of fractional knapsack problem using greedy approach***	Date: 2022-05-03

Aim:

Program to find the solution of fractional knapsack problem using greedy approach

Source Code:

knapasak.c

```
#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=1;i<=n;i++)
   {x[i]=0.0;}
   for(i=1;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("The result vector is:- \n");
      for(i=1;i<=n;i++){
         printf("%.2f\t",x[i]);}
         printf("\nMaximum profit is:- %.2f",tp);
}
int main(){int num,i,j;float w[20],p[20],c[20],cap;
float temp;
printf("Enter the no. of objects:- ");
scanf("%d",&num);
printf("Enter the Weight, Value(Profit) of each object:- \n");
for(i=1;i<=num;i++){
   printf("item %d:",i);
   scanf("%f%f",&w[i],&p[i]);}
   printf("Enter the capacity of knapsack:- ");
   scanf("%f",&cap);
   for(i=1;i<=num;i++){
      c[i]=p[i]/w[i];}
      for(i=1;i<=num;i++){
         for(j=i+1;j<=num;j++){
            if(c[i]<c[j]){
               temp=c[j];
               c[j]=c[i];
               c[i]=temp;
               temp=p[j];
               p[j]=p[i];
               p[i]=temp;
               temp=w[j];
               w[j]=w[i];
```

```
                w[i]=temp;}
            }
        }
        knapsack(num,w,p,cap);
        return 0;
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the no. of objects:- 3
Enter the Weight, Value(Profit) of each object:- 10 60
item 1: 10 60
item 2: 20 100
item 3: 30 120
Enter the capacity of knapsack:- 50
The result vector is:-
1.00 1.00 0.67
Maximum profit is:- 240.00

Test Case - 2
User Output
Enter the no. of objects:- 5
Enter the Weight, Value(Profit) of each object:- 10.0 25.0
item 1: 10.0 25.0
item 2: 10.0 25.0
item 3: 10.0 25.0
item 4: 4.0 6.0
item 5: 2.0 2.0
Enter the capacity of knapsack:- 70
The result vector is:-
1.00 1.00 1.00 1.00 1.00
Maximum profit is:- 83.00

S.No: 16	Exp. Name: ***Program to find minimum spanning tree of a given undirected graph using Kruskal's algorithm***	Date: 2022-05-04

Aim:

Program to find minimum spanning tree of a given undirected graph using Kruskal's algorithm

Source Code:

kruskalsAlgorithm.c

```
#include<stdio.h>
#include<conio.h>
int parent[100];
int find(int i)
{
   while(parent[i]!=i)
   i=parent[i];
   return i;
}
void unio(int i, int j)
{
   int x,y;
   x=find(i);
   y=find(j);
   parent[x]=y;
}
void kruskal(int a[][100],int n)
{
   int k,co=0,min,r,b,l,res[100][2];
   for(k=0;k<n;k++)
   parent[k]=k;
   printf("The minimum spanning tree has the following edges:\n");
   while(co<n-1)
   {
      min=10000000;
      r=-1;
      b=-1;
      for(k=n-1;k>-1;k--)
      {
         for(l=n-1;l>-1;l--)
         {
            if(find(k)!=find(l) && a[k][l]<min && a[k][l]!=0)
            {
               min=a[k][l];
               r=k;
               b=l;
            }
         }
      }
      unio(r,b);
      res[co][0]=r+1;
      res[co][1]=b+1;
      co++;
   }
   for(k=n-2;k>-1;k--)
   printf("%d-%d\n",res[k][0],res[k][1]);
```

```
}
void main()
{
   char c;
   int n,i,j,a[100][100],l[1000];
   printf("Input as adjacency matrix or adjacency list?(A/E)");
   scanf("%c",&c);
   printf("no of nodes :");
   scanf("%d",&n);
   printf("Input as adjacency matrix:\n");
   for(i=0;i<n;i++)
   {printf("Row %d:",i+1);
   for(j=0;j<n;j++)
   {
      scanf("%d",&a[i][j]);
   }
   }
   kruskal(a,n);
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Input as adjacency matrix or adjacency list?(A/E) A
no of nodes : 6
Input as adjacency matrix: 0 3 1 6 0 0
Row 1: 0 3 1 6 0 0
Row 2: 3 0 5 0 3 0
Row 3: 1 5 0 5 6 4
Row 4: 6 0 5 0 0 2
Row 5: 0 3 6 0 0 6
Row 6: 0 0 4 2 6 0
The minimum spanning tree has the following edges:
6-3
2-1
5-2
6-4
3-1

S.No: 17	Exp. Name: ***Program to find minimum spanning tree of a given undirected graph using Prim's Algorithm***	Date: 2022-05-04

Aim:

Program to find minimum spanning tree of a given undirected graph using Prim's Algorithm

Source Code:

primsAlgorithm.c

```
#include<stdio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[10]={ 0 },min,mincost=0,cost[10][10];
int main()
{
   printf("To compute the spanning tree from the adjacency matrix");
   printf("\nHow many nodes :");
   scanf("%d",&n);
   printf("Enter the adjacency matrix :");
   for(i=1;i<=n;i++)
   for(j=1;j<=n;j++)
   {
      scanf("%d",&cost[i][j]);
      if(cost[i][j]==0)
      cost[i][j]=999;
   }
   printf("The entered adjacency matrix :\n");
   for(i=1;i<=n;i++)
   {
      for(j=1;j<=n;j++)
      {
         if(cost[i][j]==999)
         printf("%-3d",0);
         else
         printf("%-3d",cost[i][j]);
      }
      printf("\n");
   }
   visited[1]=1;
   printf("The nodes to be connected in spanning tree are : ");
   while(ne<n)
   {
      for(i=1,min=999;i<=n;i++)
      for(j=1;j<=n;j++)
      if(cost[i][j]<min)
      if(visited[i]!=0)
      {
         min=cost[i][j];
         a=u=i;
         b=v=j;
      }
      if(visited[u]==0 || visited[v]==0)
      {
         printf("(%d,%d);",a,b);
         ne++;
         mincost+=min;
```

```
            visited[b]=1;
         }
         cost[a][b]=cost[b][a]=999;
      }
      printf("\nThe cost of Minimum Spanning Tree is :%d",mincost);
      return 0;
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
To compute the spanning tree from the adjacency matrix 5
How many nodes : 5
Enter the adjacency matrix : 0 2 0 6 0 2 0 3 8 5 0 3 0 0 7 6 8 0 0 9 0 5 7 9 0
The entered adjacency matrix :
0 2 0 6 0
2 0 3 8 5
0 3 0 0 7
6 8 0 0 9
0 5 7 9 0
The nodes to be connected in spanning tree are : (1,2);(2,3);(2,5);(1,4);
The cost of Minimum Spanning Tree is :16

Test Case - 2
User Output
To compute the spanning tree from the adjacency matrix 4
How many nodes : 4
Enter the adjacency matrix : 0 2 0 6 2 0 3 8 0 3 0 0 6 8 0 0
The entered adjacency matrix :
0 2 0 6
2 0 3 8
0 3 0 0
6 8 0 0
The nodes to be connected in spanning tree are : (1,2);(2,3);(1,4);
The cost of Minimum Spanning Tree is :11

S.No: 18	Exp. Name: ***Program to find Single source Shortest path using Dijkstra's Algorithm in weighted directed graph***	Date: 2022-05-04

Aim:

Program to find Single source Shortest path using Dijkstra's Algorithm in weighted directed graph

Source Code:

dijkstrasAlgorithm.c

```
#include<stdio.h>
#include<limits.h>
int n,k;
#define perm 1
#define tent 2
#define infinity INT_MAX
typedef struct nodelabel{
   int predecessor;
   int length;
   int label;
   int number;
}
nodelabel;
void initialize_single_source(nodelabel state[],int s,int n){
   int i;
   for(i=1;i<=n;i++){
      state[i].predecessor=0;
      state[i].length=infinity;
      state[i].label=tent;
      state[i].number=i;
   }
   state[s].predecessor=0;
   state[s].length=0;
   state[s].label=perm;
   state[s].number=s;
}
int parent(int i){
   return i/2;
}
int left(int i){
   return 2*i;
}
int right(int i){
   return 2*i+ 1;
}
void min_heapify(nodelabel q[], int i){
   struct nodelabel temp;
   int l,r,smallest;
   l=left(i);
   r=right(i);
   if(l<=k && q[l].length<q[i].length)
   smallest=l;
   else
   smallest=i;
   if(r<=k && q[r].length<q[i].length)
   smallest=r;
```

```
   if (smallest!=i){
      temp=q[i];
      q[i]=q[smallest];
      q[smallest]=temp;
      min_heapify(q, smallest);
   }
}
void build_min_heap(nodelabel q[], int n){
   int i;
   for(i=n/2; i>=1;i--)
   min_heapify(q,i);
}
nodelabel heap_extract_min(nodelabel state[]){
   nodelabel min,temp;
   min=state[1];
   temp=state[1];
   state[1]=state[k];
   state[k]=temp;
   k=k - 1;
   min_heapify(state,1);
   return min;
}
void heap_decrease_key(nodelabel state[], int key, int i){
   nodelabel temp;
   state[i].length=key;
   while(i>1 && state[parent(i)].length>state[i].length){
      temp=state[i];
      state[i]=state[parent(i)];
      state[parent(i)]=temp;
      i=parent(i);
   }
}
void relax(nodelabel u, int a[10][10],nodelabel state[],int i)
{
   int key;
   if(state[i].length>(u.length+a[u.number][state[i].number])){
      state[i].predecessor=u.number;
      key=u.length+a[u.number][state[i].number];
      heap_decrease_key(state,key,i);
   }
}
void dij(int a[][10],int n,int s){
   nodelabel state[10],min;
   int i,count,j,x,dist=0;
   int path[10];
   initialize_single_source(state,s,n);
   build_min_heap(state,n);
   while(k!=0){
      min=heap_extract_min(state);
      for(i=1;i<=k;i++)
      if(a[min.number][state[i].number]>0 && state[i].label==tent)
      relax(min,a,state,i);
      min.label=perm;
   }
   for(i=1;i<=n;i++)
   if(i!=s){
```

```
      j=i;
      dist=0;
      count=0;
      do{
         count++;
         path[count]=j;
         for(k=1;k<=n;k++)
         if(state[k].number==j){
            j=state[k].predecessor;
            break;
         }
      } while(j!=0);
      for(j=1;j<=count/2;j++){
         x=path[j];
         path[j]=path[count-j + 1];
         path[count-j + 1]=x;
      }
      for(j=1;j<count; j++)
      dist+=a[path[j]][path[j + 1]];
      printf("Shortest path from %d to %d is :",s,i);
      if(count!=1)
      printf("%d",path[1]);
      else
      printf("No path from %d to %d",s,i);
      for(j=2;j<=count;j++)
      printf("-->%d",path[j]);
      printf("\nDistance from node %d to %d is : %d",s,i,dist);
      printf("\n");
   }
}
int main(){
   int a[10][10],i,j,source;
   printf("Enter the number of nodes :");
   scanf("%d",&n);
   for(i=1;i<=n;i++){
      printf("Enter node %d connectivity :",i);
      for(j=1;j<=n;j++)
      scanf("%d", &a[i][j]);
   }
   k=n;
   printf("Enter the source node :");
   scanf("%d",& source);
   dij(a,n,source);
   return 0;
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of nodes : 3
Enter node 1 connectivity : 1 2 0
Enter node 2 connectivity : 0 5 6

Test Case - 1
Enter node 3 connectivity : 5 3 0
Enter the source node : 1
Shortest path from 1 to 2 is :1-->2
Distance from node 1 to 2 is : 2
Shortest path from 1 to 3 is :1-->2-->3
Distance from node 1 to 3 is : 8