ADA LAB – 2 SOLUTIONS

1.LCS

CODE:

```
#include <bits/stdc++.h>
using namespace std;
void lcs( char *X, char *Y, int m, int n )
{
int L[m+1][n+1];

for (int i=0; i<=m; i++)
{
        for (int j=0; j<=n; j++)
        {
        if (i == 0 || j == 0)
                L[i][j] = 0;
        else if (X[i-1] == Y[j-1])
                L[i][j] = L[i-1][j-1] + 1;
        else
                L[i][j] = max(L[i-1][j], L[i][j-1]);
        }
}

int index = L[m][n];
cout<<"Length:"<<L[m+1][n+1]<<endl;
char lcs[index+1];
lcs[index] = '\0';

int i = m, j = n;
while (i > 0 && j > 0)
{

        if (X[i-1] == Y[j-1])
        {
```

```
            lcs[index-1] = X[i-1];
            i--; j--; index--;
        }

        else if (L[i-1][j] > L[i][j-1])
            i--;
        else
            j--;
}


cout << "LCS:" << lcs<<"\n";
}


int main()
{
char X[30],Y[30];
cout<<"Enter 1st Sequence:";
cin>>X;
cout<<"Enter 2nd Sequence:";
cin>>Y;
int m = strlen(X);
int n = strlen(Y);
lcs(X, Y, m, n);
return 0;
}

/*OUTPUT:
Enter 1st Sequence:ABCDAF
Enter 2nd Sequence:ACBCF
Lenght:4
LCS:ABCF
*/
```

2. 0-1 Knapsack

CODE:

```
#include<bits/stdc++.h>
using namespace std;
int max(int a, int b)
{
        return (a > b)? a : b;
}
void knapSack(int W, int wt[], int val[], int n)
{
int i, j;
int K[n+1][W+1];
for (i = 0; i <= n; i++)
{
        for (j = 0; j <= W; j++)
        {
                if (i==0 || j==0)
                        K[i][j] = 0;
                else if (wt[i-1] <= j)
                                K[i][j] = max(val[i-1] + K[i-1][j-wt[i-1]], K[i-1][j]);
                else
                                K[i][j] = K[i-1][j];
        }
}
cout<<"\n";
  for(int i=0;i<=n;i++){
        for(int j=0;j<=W;j++){
           cout<<K[i][j]<<"\t";
        }
        cout<<"\n";
    }
int res = K[n][W];
    cout<<"\nMaximum Profit: "<<res<<endl;
    j = W;
         cout<<"Weights of items included:\n";
    for (i = n; i > 0 && res > 0; i--)
         {
```

```
        if (res == K[i - 1][j])
           continue;
        else
                {
           cout<<i<<"->"<<wt[i - 1]<<endl;
           res = res - val[i - 1];
           j= j- wt[i - 1];
        }
    }
}
int main()
{
         int val[10],wt[10],W,n;
         cout<<"Enter the no of items:";
         cin>>n;
         cout<<"\n";
         for(int i=0;i<n;i++){
                cout<<"Enter the weight "<<i+1<<":";
                cin>>wt[i];
         }
         cout<<"\n";
         for(int i=0;i<n;i++)
         {
                cout<<"Enter the values "<<i+1<<":";
                cin>>val[i];
         }
         cout<<"\nMax capacity:";
         cin>>W;
         knapSack(W, wt, val, n);
         return 0;
}

/*OUTPUT:
Enter the no of items:4
Enter the weight 1:2
Enter the weight 2:1
Enter the weight 3:3
Enter the weight 4:2
Enter the values 1:12
```

Enter the values 2:10
Enter the values 3:20
Enter the values 4:15
Max capacity:5

0	0	0	0	0	0
0	0	12	12	12	12
0	10	12	22	22	22
0	10	12	22	30	32
0	10	15	25	30	37

Maximum Profit: 37
Weights of items included:
4->2
2->1
1->2
*/

3.Minimum edit distance

CODE:

```
#include<bits/stdc++.h>
using namespace std;

int min(int x, int y, int z)
{
return min(min(x, y), z);
}

int editDist(string str1 , string str2 , int m ,int n)
{
	if (m == 0) return n;


	if (n == 0) return m;

	if (str1[m-1] == str2[n-1])
		return editDist(str1, str2, m-1, n-1);
```

```
        return 1 + min ( editDist(str1, str2, m, n-1), // Insert
                                    editDist(str1, str2, m-1, n), // Remove
                                    editDist(str1, str2, m-1, n-1) // Replace
                                );
}
int main()
{
        string str1 = "sunday";
        string str2 = "saturday";

        cout << editDist( str1 , str2 , str1.length(), str2.length());

        return 0;
}
```

4. Smallest Range

CODE:

```
#include <bits/stdc++.h>
using namespace std;

struct minHeapNode{

   int val;
   int listIdx;
   int nextListIdx;
};

void heapify(minHeapNode* harr, int k, int i){

   int smallest = i;
   int left = 2*i + 1;
   int right = 2*i + 2;

   if (left < k && harr[left].val < harr[smallest].val){
      smallest = left;
```

```
  }

  if (right < k && harr[right].val < harr[smallest].val){
    smallest = right;
  }

  if (smallest != i){
    swap(harr[i], harr[smallest]);
    heapify(harr, k, smallest);
  }
}

void buildHeap(minHeapNode* harr, int k){

  int i = k/2 - 1;
  while(i >= 0){
    heapify(harr, k, i);
    i--;
  }
}

void replace(minHeapNode* harr, int k, minHeapNode node){
  harr[0] = node;
  heapify(harr, k, 0);
}

void smallestRange(int** arr, int n, int k){

  int max = INT_MIN, min = INT_MAX;
  int range = INT_MAX;
  int start, end;

  minHeapNode* harr = new minHeapNode[k];

  for (int i = 0; i < k; i++){
    harr[i].val = arr[i][0];
    harr[i].listIdx = i;
    harr[i].nextListIdx = 1;
```

```
    if (max < harr[i].val){
      max = harr[i].val;
    }
  }

  buildHeap(harr, k);

  while(true){
    minHeapNode temp = harr[0];
    min = temp.val;
    if (range > max - min + 1){
      range  = max - min + 1;
      start = min;
      end = max;
    }

    if (temp.nextListIdx < n){
      temp.val = arr[temp.listIdx][temp.nextListIdx];
      temp.nextListIdx += 1;

      if (max < temp.val){
        max = temp.val;
      }
    }
    else{
      break;
    }
    replace(harr, k, temp);
  }

  cout << "Smallest Range : ["<< start<<" : "<< end<<"]"<< endl;
}

int main(){
  int k, n;
  cin >> k >> n;
  int arr[10][10] ;
  for(int i = 0; i < k; i++){
       for (int j = 0; j < n; j++){
```

```
        cin >> arr[i][j];
      }
    }

    smallestRange(arr, n, k);
}
```

5.Minimum Coin Exchange

CODE:

```
#include<bits/stdc++.h>
using namespace std;
#define INF 999
//#define N 3
//#define A 8
int N,A;
void coinChange(int d[], int C[], int S[]) {
  int i, p, min, coin;

  //when amount is 0
  //then min coin required is 0
  C[0] = 0;
  S[0] = 0;

  for(p = 1; p <= A; p++) {
    min = INF;
    for(i = 1; i <= N; i++) {
      if(d[i] <= p) {
        if(1 + C[p - d[i]] < min) {
          min = 1 + C[p - d[i]];
          coin = i;
        }
      }
    }
    C[p] = min;
    S[p] = coin;
```

```
  }
}

void coinSet(int d[], int S[]) {
  int a = A;
  while(a > 0) {
        cout<<"\nUse coin of denomination: "<< d[S[a]];
    a = a - d[S[a]];
  }
}

void display(int arr[]) {
  int c;
  for(c = 0; c <= A; c++) {
    printf("%5d", arr[c]);
  }
  cout<<"\n";
}
int main() {

    cout<<"Enter no.of denominations:";
    cin>>N;
    int d[N+1];
    d[0]=0;
cout<<"Enter the denominations:\n";
for(int i=1;i<=N;i++){
    cin>>d[i];
}
cout<<"Enter the amount:";
cin>>A;
  int C[A+1];
  int S[A+1];
  coinChange(d, C, S);

  cout<<"\nC[p]\n";
  display(C);

  cout<<"\nS[p]\n";
  display(S);
```

```
  cout<<"\nMin. no. of coins coin required to make change for amount "<<A<<" =
"<< C[A]<<"\n";

  cout<<"\nCoin Set\n";
  coinSet(d, S);

  return 0;
}
/*
Enter no.of denominations:3
Enter the denominations
1
2
5
Enter the amount:6
C[p]
   0   1   1   2   2   1   2
S[p]
   0   1   2   1   2   3   1
Min. no. of coins coin required to make change for amount 6 : 2

Coin Set
Use coin of denomination: 1
Use coin of denomination: 5
*/
```

6.Prims Algorithm:

CODE:

```
#include<bits/stdc++.h>
using namespace std;
int c[20][20];
void prims(int n){
    int ne=0,mini;
    int mincost=0,u,v;
    int elec[n];
```

```
    for(int i=1;i<=n;i++){
       elec[i]=1;
       }
    elec[1]=1;
    while(ne!=n-1){
       mini=9999;
       for(int i=1;i<=n;i++){
          for(int j=1;j<=n;j++){
          if(elec[j]==1){
             if(c[i][j]<mini){
                mini=c[i][j];
                u=i;
                v=j;
             }
          }
       }
    }
    if(elec[v]==1){
       cout<<u<<"-->"<<v<<"="<<mini<<"\n";
       elec[v]=0;
       ne=ne+1;
       mincost=mincost+mini;
    }
    c[u][v]=9999;
    c[v][u]=9999;
    }
    cout<<"mincost="<<mincost;

}
int main(){
   int n;
   cout<<"enter no. of vertices";
   cin>>n;
   cout<<"\nenter cost matrix\n";
   for(int i=1;i<=n;i++){
    for(int j=1;j<=n;j++){
       cin>>c[i][j];
    }
   }
```

```
   prims(n);
   return 0;
}
/*
enter no. of vertices4
enter cost matrix
9999 3 4 1
3 9999 2 9999
4 2 9999 9999
1 9999 9999 9999
1-->4=1
2-->3=2
1-->2=3
mincost=6
*/
```

7.Partition Problem

CODE:

```
#include <bits/stdc++.h>
using namespace std;

bool isSubsetSum (int arr[], int n, int sum)
{
        if (sum == 0)
                return true;
        if (n == 0 && sum != 0)
                return false;

        if (arr[n-1] > sum)
        return isSubsetSum (arr, n-1, sum);
        return isSubsetSum (arr, n-1, sum) ||
                isSubsetSum (arr, n-1, sum-arr[n-1]);
}

bool findPartiion (int arr[], int n)
{
        int sum = 0;
```

```
    for (int i = 0; i < n; i++)
    sum += arr[i];

    if (sum%2 != 0)
    return false;

    return isSubsetSum (arr, n, sum/2);
}

int main()
{
    int arr[] = {3, 1, 5, 9, 12};
    int n = sizeof(arr)/sizeof(arr[0]);
    if (findPartiion(arr, n) == true)
        cout << "Can be divided into two subsets of equal sum";
    else
        cout << "Can not be divided into two subsets of equal sum";
    return 0;
}
```

8. Dijkstra's

CODE:

```
#include<bits/stdc++.h>
using namespace std;
int n;
int dijkstra(int c[][],int src)
{
    int dist[20];
    bool vis[20];
    int count=0,min,u;
    for(int j=0;j<n;j++)
    {
        dist[j]=c[src][j];
    }
    for(int j=0;j<n;j++)
    {
        vis[j]=false;
```

```
        }
        dist[src]=0;
        vis[src]=true;
        count=1;
        while(count!=(n-1))
        {
                min = INT_MAX;
                for(int j=0;j<n;j++)
                {
                        if((dist[j]<min)&&(!vis[j]))
                        {
                                min=dist[j];
                                u=j;
                        }
                }
                vis[u]=true;
                count+=1;
                for(int j=0;j<n;j++)
                {
                        if((min+c[u][j])<dist[j] && (!vis[j]))
                        {
                                dist[j]=min+c[u][j];
                        }
                }
        }
        cout<<"The shortest distance are : "<<endl;
        for(int j=0;j<n;j++)
        {
                cout<<src<<" -> "<<j<<" = "<<dist[j]<<endl;
        }
        return 0;
}
int main()
{
        int adjmat[20][20];
        int src;
        cout<<"Enter the number of vertices : ";
        cin>>n;
        cout<<n<<endl;
```

```
	cout<<"Enter the adjacency matrix : "<<endl;
	for(int i=0;i<n;i++)
	{
		for(int j=0;j<n;j++)
		{
			cin>>adjmat[i][j];
			cout<<adjmat[i][j]<<" ";
		}
		cout<<endl;
	}
	cout<<"Enter the source vertex : ";
	cin>>src;
	cout<<src<<endl;
	dijkstra(adjmat,src);
}
```

9. Floyd's Algorithm

CODE:

```
#include<bits/stdc++.h>
#include<algorithm>
using namespace std;
int n;
int floyd(int a[][])
{
	for(int k=0;k<n;k++)
	{
		for(int i=0;i<n;i++)
		{
			for(int j=0;j<n;j++)
			{
				int mn = a[i][k]+a[k][j];
				if(mn < a[i][j])
				{
					a[i][j]=mn;
				}
			}
		}
```

```
        }
        cout<<"The resultant matrix is "<<endl;
        for(int i=0;i<n;i++)
        {
                for(int j=0;j<n;j++)
                {
                        cout<<a[i][j]<<" ";
                }
                cout<<endl;
        }
        return 0;
}
int main()
{
        int adjmat[20][20];
        cout<<"Enter the number of Vertices : ";
        cin>>n;
        cout<<n<<endl;
        cout<<"Enter the adjacent matrix "<<endl;
        for(int i=0;i<n;i++)
        {
                for(int j=0;j<n;j++)
                {
                        cin>>adjmat[i][j];
                        cout<<adjmat[i][j]<<" ";
                }
                cout<<endl;
        }
        floyd(adjmat);
        return 0;
}
```

10.Longest increasing subsequence

CODE:

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

```
int _lis( int arr[], int n, int *max_ref)
{
	if (n == 1)
		return 1;

	int res, max_ending_here = 1;


	for (int i = 1; i < n; i++)
	{
		res = _lis(arr, i, max_ref);
		if (arr[i-1] < arr[n-1] && res + 1 > max_ending_here)
			max_ending_here = res + 1;
	}


	if (*max_ref < max_ending_here)
	*max_ref = max_ending_here;

	return max_ending_here;
}

int lis(int arr[], int n)
{
	int max = 1;
	_lis( arr, n, &max );

	return max;
}
int main()
{
	int arr[] = { 10, 22, 9, 33, 21, 50, 41, 60 };
	int n = sizeof(arr)/sizeof(arr[0]);
	printf("Length of lis is %dn",
		lis( arr, n ));
	return 0;
}
```