Lab:-9

Perform Knapsack problem using Dynamic programming technique using n=4 objects with associated weights and profits .
Display the table values and the objects selected in the knapsack to get maximum profit.

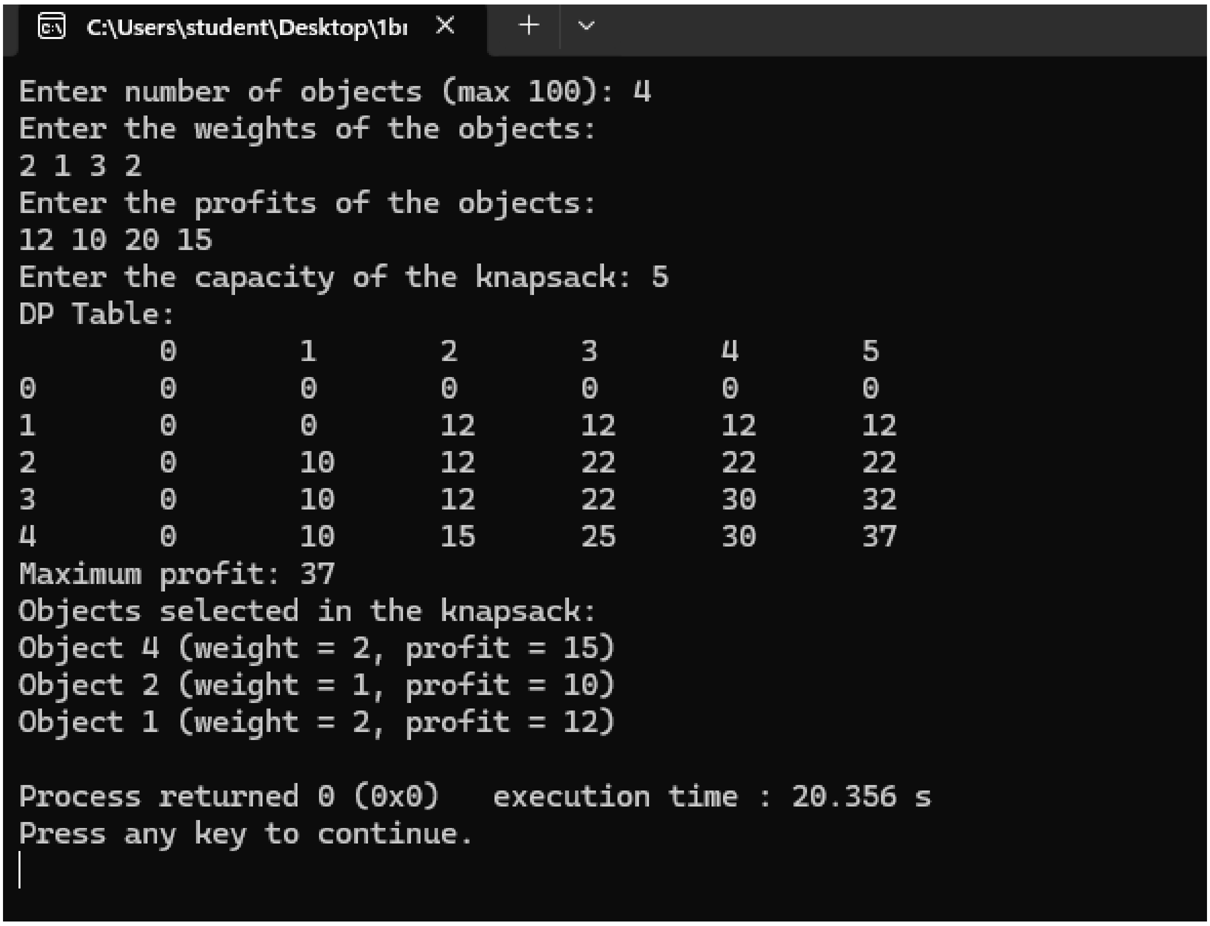
Code:-

```
#include <stdio.h>

#define MAX_OBJECTS 100

// Function to calculate maximum of two integers
int max(int a, int b) {
    return (a > b) ? a : b;
}

// Function to solve 0/1 Knapsack problem using Dynamic Programming
void knapsack(int n, int W, int weights[], int profits[]) {
    int i, w;
    int K[MAX_OBJECTS + 1][W + 1]; // DP table to store results

    // Build DP table K[][] in bottom-up manner
    for (i = 0; i <= n; i++) {
        for (w = 0; w <= W; w++) {
            if (i == 0 || w == 0)
                K[i][w] = 0;
            else if (weights[i - 1] <= w)
                K[i][w] = max(profits[i - 1] + K[i - 1][w - weights[i - 1]], K[i - 1][w]);
            else
                K[i][w] = K[i - 1][w];
        }
    }

    // Print DP table with indices
    printf("DP Table:\n");
    printf("\t");
    for (w = 0; w <= W; w++) {
        printf("%d\t", w);
    }
    printf("\n");
    for (i = 0; i <= n; i++) {
        printf("%d\t", i);
        for (w = 0; w <= W; w++) {
            printf("%d\t", K[i][w]);
        }
        printf("\n");
    }

    // Maximum profit will be at K[n][W]
    int maxProfit = K[n][W];
    printf("Maximum profit: %d\n", maxProfit);
```

```
    // To find the selected items
    printf("Objects selected in the knapsack:\n");
    int res = maxProfit;
    w = W;
    for (i = n; i > 0 && res > 0; i--) {
        if (res == K[i - 1][w])
            continue;
        else {
            printf("Object %d (weight = %d, profit = %d)\n", i, weights[i - 1], profits[i - 1]);
            // Move to the previous item considering its weight
            res -= profits[i - 1];
            w -= weights[i - 1];
        }
    }
}

int main() {
    int n, W;
    int weights[MAX_OBJECTS], profits[MAX_OBJECTS];
    int i;

    // Input number of objects
    printf("Enter number of objects (max %d): ", MAX_OBJECTS);
    scanf("%d", &n);

    if (n <= 0 || n > MAX_OBJECTS) {
        printf("Invalid number of objects\n");
        return 1;
    }

    // Input weights of objects
    printf("Enter the weights of the objects:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &weights[i]);
    }

    // Input profits of objects
    printf("Enter the profits of the objects:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &profits[i]);
    }

    // Input knapsack capacity
    printf("Enter the capacity of the knapsack: ");
    scanf("%d", &W);

    if (W <= 0) {
        printf("Invalid knapsack capacity\n");
        return 1;
```

```
    }

    knapsack(n, W, weights, profits);

    return 0;
}
```

Output:-



```
C:\Users\student\Desktop\1b
Enter number of objects (max 100): 4
Enter the weights of the objects:
2 1 3 2
Enter the profits of the objects:
12 10 20 15
Enter the capacity of the knapsack: 5
DP Table:
        0       1       2       3       4       5
0       0       0       0       0       0       0
1       0       0       12      12      12      12
2       0       10      12      22      22      22
3       0       10      12      22      30      32
4       0       10      15      25      30      37
Maximum profit: 37
Objects selected in the knapsack:
Object 4 (weight = 2, profit = 15)
Object 2 (weight = 1, profit = 10)
Object 1 (weight = 2, profit = 12)

Process returned 0 (0x0)   execution time : 20.356 s
Press any key to continue.
```

Pfa of the Prims algorithm pseudo code please try to convert this into C program and find the MST of a Given graph with cost adjacency matrix as input.

Algorithm :-

Algorithm Prims(n,cost)

Purpose: To compute the Minimum Spanning Tree

//Input: n number of vertices in the graph

Cost : Cost adjacency matrix with values >0

//Output : d- shortest distance from source to all other nodes.

p- Shortest path from source to destination

s- gives the information nodes that are so far visited and the nodes that are not visted.

Step 1: [Obtain a source vertex which has the least edge going out of it]

Min⍰ 9999; Source⍰0

For i<-0 to n-1

For j<- 0 to n-1

If(cost[i,j]!=0 && cost[i,j]<min)

Min=cost[i][j]

Source=i

End if.

Step 2: [Initialization]

For i<-0 to n-1 do

S[i]=0, d[i]=cost[Source,i]

P[i]=source

End for

Step 3: {Add Source to s]

S[source]=1

Step 4: [Find the Minimum spanning tree if exists]

Sum<-0; k<-0

For i<-1 to n-1 do

// find u and d[u] such that d[u] is minimum and u Є v-s

Min⍰9999

U=-1

For j <- 0 to n-1 do

If(s[j]=0 and d[j] <=min)

Min<-d[j]

U<-j

End if

End for

//Select an edge with the least cost

T[K][0]<- U T[K][1]<-P[U] K<-K+1

//Add the cost associated with the edge to get total cost of MST.

Sum<-sum + cost[u][p[u]]

//Add u to s

S[u]<- 1

//Find the new vertex u and distance which gives the shortest path and destination.

For every v Є v –s do

If(cost[u][v] < d[v])

D[v]=cost[u][v]

P[v]=u

End if

End for

End for // Outer for Loop

Step 5: [Check for the existence of spanning tree]

If(sum >=9999)

Write “spanning tree does not exist”

Else

Write “Spanning tree exists and MST is”

For i<-0 to n-2 do

Write T[i][0], T[i][1]

End for

Write “The cost of Spanning tree is MST is”, sum

End if

Code:-

```
#include <stdio.h>

#include <string.h>

#include <limits.h>


#define MAX_VERTICES 100
```

```
#define INF INT_MAX

int minKey(int n, int d[], int s[]) {
    int min = INF, min_index;

    for (int v = 0; v < n; v++) {
        if (s[v] == 0 && d[v] < min) {
            min = d[v];
            min_index = v;
        }
    }

    return min_index;
}
int printMST(int n, int p[], int cost[MAX_VERTICES][MAX_VERTICES]) {
    int total_cost = 0;
    printf("Edge   Weight\n");
    for (int i = 1; i < n; i++) {
        printf("%d - %d    %d \n", p[i], i, cost[i][p[i]]);
        total_cost += cost[i][p[i]];
    }
    return total_cost;
}

int parseCost(int n, int cost[MAX_VERTICES][MAX_VERTICES]) {
    char input[10];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%s", input);
            if (strcmp(input, "inf") == 0) {
                cost[i][j] = INF;
```

```
            } else {
                sscanf(input, "%d", &cost[i][j]);
                if (cost[i][j] == 0 && i != j) {
                    cost[i][j] = INF;
                }
            }
        }
    }
}

void primMST(int n, int cost[MAX_VERTICES][MAX_VERTICES]) {
    int p[MAX_VERTICES];
    int d[MAX_VERTICES];
    int s[MAX_VERTICES];
    for (int i = 0; i < n; i++) {
        d[i] = INF;
        s[i] = 0;
    }

    d[0] = 0;
    p[0] = -1;

    for (int count = 0; count < n - 1; count++) {
        int u = minKey(n, d, s);

        s[u] = 1;
        for (int v = 0; v < n; v++) {
            if (cost[u][v] && s[v] == 0 && cost[u][v] < d[v]) {
                p[v] = u;
                d[v] = cost[u][v];
            }
```

```c
        }
    }
    int total_cost = printMST(n, p, cost);
    printf("Total cost of Minimum Spanning Tree (MST): %d\n", total_cost);
}

int main() {
    int n;
    int cost[MAX_VERTICES][MAX_VERTICES];
    printf("Enter number of vertices (max %d): ", MAX_VERTICES);
    scanf("%d", &n);
    printf("Enter the cost adjacency matrix (use 'inf' for infinity):\n");
    parseCost(n, cost);
    printf("Minimum Spanning Tree (MST) using Prim's algorithm:\n");
    primMST(n, cost);

    return 0;
}
```

Output:-

C:\Users\student\Desktop\1b

```
Enter number of vertices (max 100): 6
Enter the cost adjacency matrix (use 'inf' for infinity):
0 60 10 inf inf inf
60 0 inf 20 40 70
10 inf 0 inf inf 50
inf 20 inf 0 inf 80
inf 40 inf inf 0 30
inf 70 50 80 30 0
Minimum Spanning Tree (MST) using Prim's algorithm:
Edge   Weight
4 - 1    40
0 - 2    10
1 - 3    20
5 - 4    30
2 - 5    50
Total cost of Minimum Spanning Tree (MST): 150

Process returned 0 (0x0)   execution time : 70.720 s
Press any key to continue.
```

return 0; }

}

Output:-

Enter the number of vertices : 6

Enter the cost of adjacency matrix

(use inf for infinity);

0 60 10 inf inf inf

60 0 inf 20 40 70

10 inf 0 inf inf 50

inf 20 inf 0 inf 50.

inf 40 inf inf 0 30

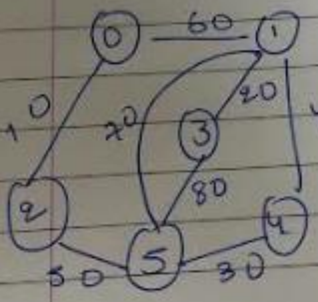
inf 70 50 50 30 0

Minimum Spanning Tree (MST)

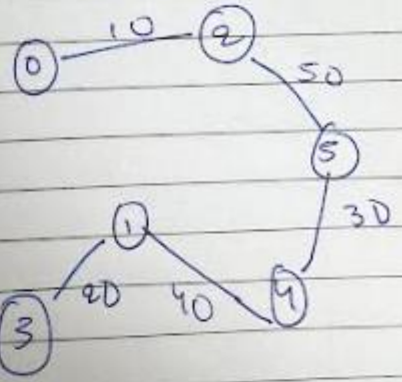
using Prim's algorithm:-

Edge	weight
4 - 1	40
0 - 2	10
1 - 3	20
5 - 4	30
2 - 5	50.

Total cost of Minimum Spanning Tree : 150



MST.



total cost : 150.