Date of the Session:__/__/_____ **Time of the Session:_____to_____**

EX – 6 Working with Job Sequencing, Huffman Coding, Single Source Shortest Path and Minimum Cost Spanning Trees.

Prerequisites:

- Basics of Data Structures and C Programming.
- Basic knowledge about arrays, trees and graphs.

Pre-Lab:

1) Our TASC has the information that a scientist has gone crazy and was planning a gas leakage in Delhi. We do not know the exact location where he was planning to do this attack. When Srikanth Tiwari tried to catch him, The Scientist tried to kill himself and has gone into coma in this process. Now no one knows the location of the attack except he has kept all the information in a file in his private server, but it is encoded using Huffman coding. So, help NIA decode the information. Given the root of the graph and the encoded information write a function to decode it. The function will have input parameters of root node and encoded string.
   Input

   Encoded String-1001011

   Output

   ABACA

Source code:

```
#include <stdio.h>

#include <stdlib.h>


typedef struct Node {

   char data;

   struct Node *left, *right;

} Node;
```

```
void decodeHuffman(Node *root, const char *encoded) {
    Node *curr = root;
    while (*encoded) {
        curr = (*encoded == '0') ? curr->left : curr->right;
        if (!curr->left && !curr->right) {
            printf("%c", curr->data);
            curr = root;
        }
        encoded++;
    }
    printf("\n");
}

Node *newNode(char data) {
    Node *node = (Node *)malloc(sizeof(Node));
    node->data = data;
    node->left = node->right = NULL;
    return node;
}

int main() {
    Node *root = newNode('\0');
    root->left = newNode('A');
    root->right = newNode('\0');
```

```
    root->right->left = newNode('B');
    root->right->right = newNode('C');

    char encoded[] = "1001011";
    decodeHuffman(root, encoded);

    return 0;
}
```

In-Lab:

1) You are a Human resource manager working in a Startup. You are tasked with to utilize the best of the working professionals to get the maximum profit for different jobs of a Project.
An array of jobs is given where every job has a deadline and associated profit if the job is finished before the deadline. It is also given that every job takes single unit of time, so the minimum possible deadline for any job is 1.How to maximize total profit if only one job can be scheduled at a time. Write a code for the following problem.

Input

7
a b c d e f g
3 4 4 2 3 1 2
35 30 25 20 15 12 5

Output

110
d c a b

Source code:

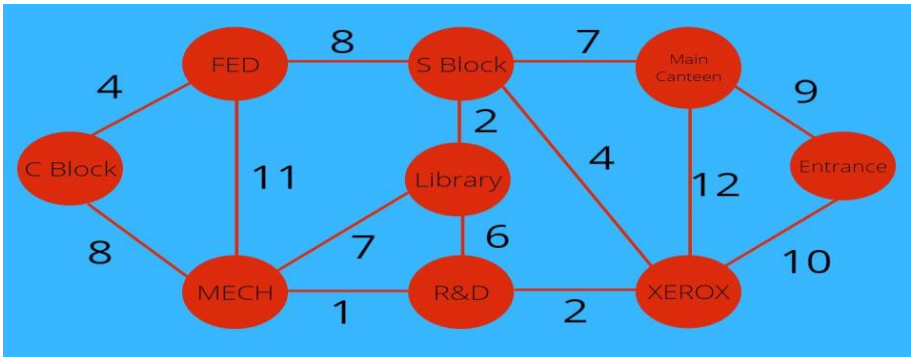
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
   char id;
   int deadline;
   int profit;
} Job;

int compare(const void *a, const void *b) {
   return ((Job *)b)->profit - ((Job *)a)->profit;
}

void job_scheduling(char ids[], int deadlines[], int profits[], int n) {
   Job jobs[n];
   for (int i = 0; i < n; i++) {
      jobs[i].id = ids[i];
      jobs[i].deadline = deadlines[i];
      jobs[i].profit = profits[i];
   }

   qsort(jobs, n, sizeof(Job), compare);

   int max_deadline = 0;
   for (int i = 0; i < n; i++) {
      if (jobs[i].deadline > max_deadline) {
         max_deadline = jobs[i].deadline;
```

```
        }
    }

    char slots[max_deadline];
    memset(slots, '-', sizeof(slots));
    int total_profit = 0;

    for (int i = 0; i < n; i++) {
        for (int j = jobs[i].deadline - 1; j >= 0; j--) {
            if (slots[j] == '-') {
                slots[j] = jobs[i].id;
                total_profit += jobs[i].profit;
                break;
            }
        }
    }

    printf("%d\n", total_profit);
    for (int i = 0; i < max_deadline; i++) {
        if (slots[i] != '-') {
            printf("%c ", slots[i]);
        }
    }
    printf("\n");
}

int main() {
    char ids[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g'};
    int deadlines[] = {3, 4, 4, 2, 3, 1, 2};
    int profits[] = {35, 30, 25, 20, 15, 12, 5};
    int n = sizeof(ids) / sizeof(ids[0]);

    job_scheduling(ids, deadlines, profits, n);

    return 0;
}
```

[@KLWKS_BOT] THANOS

2) Surya is a student at KL University, He is currently standing in the C-Block. He has 8 friends who are situated at different Blocks (Places) inside the university and he wishes to talk to each of them in person. The distances are represented on the undirected graph which is provided below. He wishes to take the shortest distance for each place from his location. Help him in meeting every one of his friends by developing a program that can determine the shortest distance between the C-Block and every other place on the graph. Remember that the path is not required.

Hint: Use Dijkstra's algorithm to calculate the distances between the C-Block and every other place
Output for the same is provided along with the question.



Output:

Vertex	Distance from Source
C Block	0
FED Block	4
S Block	12
Main Canteen	19
Entrance	21
Xerox	11
R&D Block	9
Mech Block	8
Library	14

Source code:

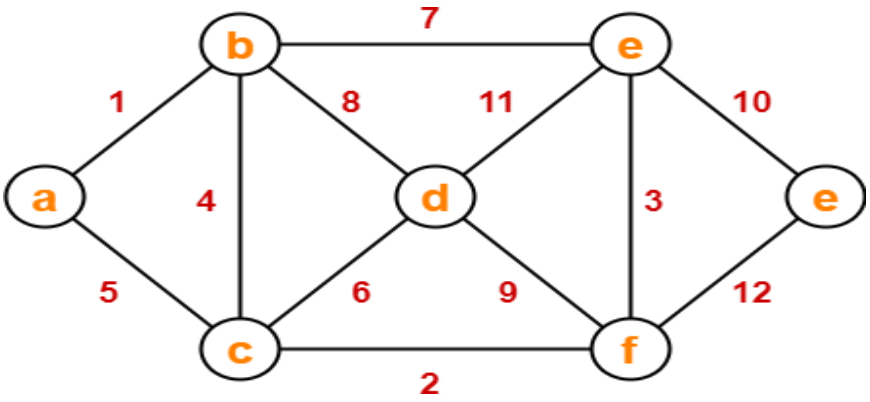
```
#include <stdio.h>
#include <limits.h>

#define V 9

int minDistance(int dist[], int sptSet[]) {
   int min = INT_MAX, min_index;

   for (int v = 0; v < V; v++) {
      if (sptSet[v] == 0 && dist[v] <= min) {
         min = dist[v], min_index = v;
      }
   }
```

```
    return min_index;
}

void printSolution(int dist[]) {
    printf("Vertex\tDistance from Source\n");
    const char* vertices[V] = {"C Block", "FED Block", "S Block", "Main Canteen", "Entrance",
"Xerox", "R&D Block", "Mech Block", "Library"};
    for (int i = 0; i < V; i++) {
        printf("%s\t%d\n", vertices[i], dist[i]);
    }
}

void dijkstra(int graph[V][V], int src) {
    int dist[V];
    int sptSet[V];

    for (int i = 0; i < V; i++) {
        dist[i] = INT_MAX;
        sptSet[i] = 0;
    }

    dist[src] = 0;

    for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, sptSet);

        sptSet[u] = 1;

        for (int v = 0; v < V; v++) {
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }

    printSolution(dist);
}
```

```
int main() {
   int graph[V][V] = {
      {0, 4, 0, 0, 0, 0, 0, 8, 0},
      {4, 0, 8, 0, 0, 0, 0, 11, 0},
      {0, 8, 0, 7, 0, 4, 0, 0, 2},
      {0, 0, 7, 0, 9, 14, 0, 0, 0},
      {0, 0, 0, 9, 0, 10, 0, 0, 0},
      {0, 0, 4, 14, 10, 0, 2, 0, 0},
      {0, 0, 0, 0, 0, 2, 0, 1, 6},
      {8, 11, 0, 0, 0, 0, 1, 0, 7},
      {0, 0, 2, 0, 0, 0, 6, 7, 0}
   };

   dijkstra(graph, 0);

   return 0;
}
```

Post-Lab:

1) Mr. Tripathiis a network cable operator. He now shifted to a new city where he needs to work on designing the cable lines to each street/colony. Given the graph and each node represents a street, help him to design an optimal cable line using prim's algorithm. Write a program to solve this.

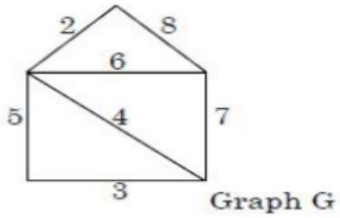


Source code:

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

#define V 6

int minKey(int key[], bool mstSet[]) {
   int min = INT_MAX, min_index;
   for (int v = 0; v < V; v++) {
      if (!mstSet[v] && key[v] < min) {
         min = key[v], min_index = v;
      }
   }
   return min_index;
}

void printMST(int parent[], int graph[V][V]) {
   printf("Minimum Spanning Tree:\n");
   for (int i = 1; i < V; i++) {
      printf("%c - %c : %d\n", 'a' + parent[i], 'a' + i, graph[i][parent[i]]);
   }
}

void primMST(int graph[V][V]) {
   int parent[V], 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] && graph[u][v] < key[v]) {
                parent[v] = u, key[v] = graph[u][v];
            }
        }
    }
    printMST(parent, graph);
}

int main() {
    int graph[V][V] = {
        {0, 1, 5, 0, 0, 0},
        {1, 0, 4, 8, 7, 0},
        {5, 4, 0, 6, 0, 2},
        {0, 8, 6, 0, 11, 9},
        {0, 7, 0, 11, 0, 3},
        {0, 0, 2, 9, 3, 0}
    };
    primMST(graph);
    return 0;
}
```

2) Mr.Tripathi done with designing the cable lines but now there comes a new task, that is working on street lines. Help him again to find weight of the graph using Kruskal algorithm. Write a program to solve this.



Graph G

Source code:

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

#define VERTICES 5
#define EDGES 7

typedef struct {
   int u, v, weight;
} Edge;

typedef struct {
   int parent[VERTICES];
   int rank[VERTICES];
} DisjointSet;

void initializeSet(DisjointSet *ds) {
   for (int i = 0; i < VERTICES; i++) {
      ds->parent[i] = i;
      ds->rank[i] = 0;
   }
}

int find(DisjointSet *ds, int u) {
   if (ds->parent[u] != u) {
      ds->parent[u] = find(ds, ds->parent[u]);
   }
   return ds->parent[u];
}

void unionSet(DisjointSet *ds, int u, int v) {
```

```
    int rootU = find(ds, u);
    int rootV = find(ds, v);

    if (rootU != rootV) {
        if (ds->rank[rootU] > ds->rank[rootV]) {
            ds->parent[rootV] = rootU;
        } else if (ds->rank[rootU] < ds->rank[rootV]) {
            ds->parent[rootU] = rootV;
        } else {
            ds->parent[rootV] = rootU;
            ds->rank[rootU]++;
        }
    }
}

int compareEdges(const void *a, const void *b) {
    return ((Edge*)a)->weight - ((Edge*)b)->weight;
}

int kruskalMST(Edge edges[]) {
    qsort(edges, EDGES, sizeof(Edge), compareEdges);

    DisjointSet ds;
    initializeSet(&ds);

    int mst_weight = 0, count = 0;

    for (int i = 0; i < EDGES; i++) {
        if (find(&ds, edges[i].u) != find(&ds, edges[i].v)) {
            unionSet(&ds, edges[i].u, edges[i].v);
            mst_weight += edges[i].weight;
            count++;
            if (count == VERTICES - 1) break;
        }
    }
    return mst_weight;
}
```

```
int main() {
   Edge edges[EDGES] = {
      {0, 1, 2}, {1, 2, 8}, {1, 3, 6}, {2, 3, 4},
      {2, 4, 5}, {3, 4, 3}, {0, 2, 7}
   };


   printf("Weight of MST: %d\n", kruskalMST(edges));
   return 0;
}
```

Comments of the Evaluators (if Any)	Evaluator's Observation Marks Secured:_____out of [50]. Signature of the Evaluator Date of Evaluation: