Design And Analysis of Algorithm

Name: Harsh Brahmecha
PRN: 20220802003
LAB 8

1) Write a code to implement Prims Algorithm

```
#include <stdio.h>
#include <stdbool.h> // Include the header for bool type
#include <limits.h>

#define V 5

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

void printMST(int parent[], int graph[V][V]) {
    printf("Edge \tWeight\n");
    for (int i = 1; i < V; i++) {
        printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
    }
}

void primMST(int graph[V][V]) {
    int parent[V];
    int key[V];
    bool mstSet[V];

    for (int i = 0; i < V; i++) {
        key[i] = INT_MAX;
        mstSet[i] = false;
    }

    key[0] = 0;
```

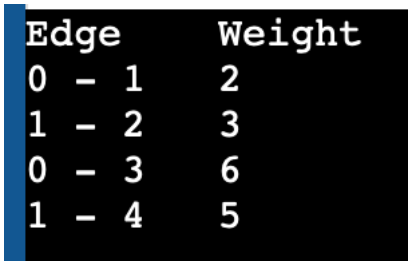
```
    parent[0] = -1;

    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] == false && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][v];
            }
        }
    }
    printMST(parent, graph);
}

int main() {
    int graph[V][V] = {{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}};

    primMST(graph);
    return 0;
}
```

Output:



2) Write a code to implement Rabin-Karp Algorithm

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

#define d 256 // Number of characters in the input alphabet

void search(char pattern[], char text[], int q) {
    int M = strlen(pattern);
    int N = strlen(text);
    int i, j;
    int p = 0; // hash value for pattern
    int t = 0; // hash value for text
    int h = 1;

    // The value of h would be "pow(d, M-1)%q"
    for (i = 0; i < M - 1; i++) {
        h = (h * d) % q;
    }

    // Calculate the hash value of pattern and first window of text
    for (i = 0; i < M; i++) {
        p = (d * p + pattern[i]) % q;
        t = (d * t + text[i]) % q;
    }

    // Slide the pattern over text one by one
    for (i = 0; i <= N - M; i++) {
        // Check the hash values of current window of text and pattern.
        // If the hash values match, then only check for characters one by one
        if (p == t) {
            // Check for characters one by one
            for (j = 0; j < M; j++) {
                if (text[i + j] != pattern[j])
                    break;
            }

            // If pattern[0...M-1] = text[i, i+1, ...i+M-1]
            if (j == M) {
                printf("Pattern found at index %d\n", i);
            }
        }
```

```
        // Calculate hash value for next window of text: Remove leading digit,
        // add trailing digit
        if (i < N - M) {
            t = (d * (t - text[i] * h) + text[i + M]) % q;

            // We might get negative value of t, converting it to positive
            if (t < 0)
                t = (t + q);
        }
    }
}

int main() {
    char text[] = "AABAACAADAABAAABAA";
    char pattern[] = "AABA";
    int q = 101; // A prime number

    search(pattern, text, q);
    return 0;
}
```

Output: