Name:- Yash Sanjay Kale
Reg NO:- 2020BIT047

Practical No:-04 Write a C/C++ code for following Algorithm with explanation

1. 1) Travelling salesman Problem.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <limits>

using namespace std;

const double INF = numeric_limits<double>::infinity();

int nearest_neighbor(int n, vector<vector<double>> &distances) {
    vector<bool> visited(n, false);
    visited[0] = true; // start at city 0
    int current_city = 0;
    int next_city;
    double total_distance = 0;
    for (int i = 0; i < n-1; i++) {
        double min_distance = INF;
        for (int j = 0; j < n; j++) {
            if (!visited[j] && distances[current_city][j] < min_distance) {
                min_distance = distances[current_city][j];
                next_city = j;
            }
        }
        visited[next_city] = true;
        current_city = next_city;
        total_distance += min_distance;
    }
    total_distance += distances[current_city][0]; // return to starting city
    return total_distance;
}

int main() {
    int n; // number of cities
    cin >> n;
    vector<vector<double>> distances(n, vector<double>(n));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> distances[i][j];
        }
    }
    cout << nearest_neighbor(n, distances) << endl;
    return 0;
}
```

2.BF string Matching Algorithm

```
#include <iostream>
#include <string>

using namespace std;
```

```
int bf_search(string text, string pattern) {
    int n = text.length();
    int m = pattern.length();
    for (int i = 0; i <= n-m; i++) {
        int j;
        for (j = 0; j < m; j++) {
            if (text[i+j] != pattern[j]) {
                break;
            }
        }
        if (j == m) { // match found
            return i;
        }
    }
    return -1; // match not found
}

int main() {
    string text, pattern;
    cin >> text >> pattern;
    int index = bf_search(text, pattern);
    if (index != -1) {
        cout << "Match found at index " << index << endl;
    } else {
        cout << "Match not found" << endl;
    }
    return 0;
}
```

3Exhaustive Search Algorithm

```
#include <iostream>
#include <vector>

using namespace std;

bool subset_sum(int target_sum, vector<int> &numbers) {
    int n = numbers.size();
    for (int i = 0; i < (1<<n); i++) { // iterate over all subsets of numbers
        int subset_sum = 0;
        for (int j = 0; j < n; j++) {
            if (i & (1<<j)) { // check if jth bit is set in i
                subset_sum += numbers[j];
            }
        }
        if (subset_sum == target_sum) { // subset with target sum found
            return true;
        }
    }
    return false; // no subset with target sum found
}

int main() {
    int target_sum, n;
    cin >> target_sum >> n;
    vector<int> numbers(n);
    for (int i = 0; i < n; i++) {
        cin >> numbers[i];
    }
    if (subset_sum(target_sum, numbers)) {
        cout << "There exists a subset with the target sum" << endl;
    } else {
        cout << "There does not exist a subset with the target sum" << endl;
    }
    return 0;
}
```