ASSIGNMENT 7

1)

```
#include <iostream>
#include <vector>
using namespace std;
// Brute Force Pattern Matching
void bruteForce(string txt, string pat) {
    int N = txt.length();
    int M = pat.length();
    cout << "Brute Force Output:\n";
    for (int i = 0; i <= N - M; i++) {
        int j;
        for (j = 0; j < M; j++) {
            if (txt[i + j] != pat[j])
                break;
        }
        if (j == M)
            cout << "Pattern found at index " << i + 1 << endl;
    }
}
// Rabin-Karp Pattern Matching
void rabinKarp(string txt, string pat, int q = 101) {
    int d = 256; // Number of characters in input alphabet
    int N = txt.length();
    int M = pat.length();
    int p = 0;  // hash value for pattern
    int t = 0;  // hash value for text
    int h = 1;

    cout << "Rabin-Karp Output:\n";
    // h = pow(d, M-1) % q
    for (int i = 0; i < M - 1; i++)
        h = (h * d) % q;
    // Calculate the hash value of pattern and first window
    for (int i = 0; i < M; i++) {
        p = (d * p + pat[i]) % q;
        t = (d * t + txt[i]) % q;
    }

    for (int i = 0; i <= N - M; i++) {
        if (p == t) {
            bool match = true;
            for (int j = 0; j < M; j++) {
                if (txt[i + j] != pat[j]) {
                    match = false;
                    break;
                }
            }
            if (match)
                cout << "Pattern found at index " << i + 1 << endl;
        }

        if (i < N - M) {
            t = (d * (t - txt[i] * h) + txt[i + M]) % q;
            if (t < 0)
                t = (t + q);
        }
    }
}
// KMP Pattern Matching
void computeLPSArray(string pat, vector<int>& lps) {
    int length = 0;
    int i = 1;
    lps[0] = 0;

    while (i < pat.length()) {
        if (pat[i] == pat[length]) {
            length++;
            lps[i] = length;
            i++;
        } else {
            if (length != 0)
                length = lps[length - 1];
            else {
                lps[i] = 0;
                i++;
            }
        }
    }
}

void KMP(string txt, string pat) {
    int N = txt.length();
    int M = pat.length();
    vector<int> lps(M);
    computeLPSArray(pat, lps);

    cout << "KMP Output:\n";
    int i = 0, j = 0;
    while (i < N) {
        if (pat[j] == txt[i]) {
            i++; j++;
        }
        if (j == M) {
            cout << "Pattern found at index " << i - j + 1 << endl;
            j = lps[j - 1];
        } else if (i < N && pat[j] != txt[i]) {
            if (j != 0)
                j = lps[j - 1];
            else
                i++;
        }
    }
}

// Main function to run all three algorithms
int main() {
    string txt1 = "THIS IS A TEST TEXT";
    string pat1 = "TEST";
    string txt2 = "AABAACAADAABAABA";
    string pat2 = "AABA";

    // Run all algorithms on first input
    cout << "First Test Case:\n";
    bruteForce(txt1, pat1);
    rabinKarp(txt1, pat1);
    KMP(txt1, pat1);

    cout << "\nSecond Test Case:\n";
    bruteForce(txt2, pat2);
    rabinKarp(txt2, pat2);
    KMP(txt2, pat2);

    return 0;
}
```