**DAA**

**CSE2012**

**LAB-08**

**Name: Preyash**

**Registration Number: 20BPS1022**

**String matching Naive approach**

**Code:**

#include <bits/stdc++.h>

using namespace std;

void search(char*\** pat, char*\** txt)

{

 int M = strlen(pat);

 int N = strlen(txt);

 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 << endl;

    }

}

int main()

{

    char txt[] = "AABAACAADAABAAABAA";

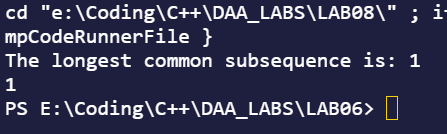
    char pat[] = "AABA";

    search(pat, txt);

    return 0;

}

**Output:**

****

**Knuth Morris Pratt Pattern Searching**

**Code:**

#include <bits/stdc++.h>

void computeLPSArray(char*\** pat, int M, int*\** lps);

void KMPSearch(char*\** pat, char*\** txt)

{

    int M = strlen(pat);

    int N = strlen(txt);

    int lps[M];

    computeLPSArray(pat, M, lps);

    int i = 0; *// index for txt[]*

    int j = 0; *// index for pat[]*

    while (i < N) {

        if (pat[j] == txt[i]) {

        j++;

        i++;

    }

    if (j == M) {

    printf("Pattern found at index %d ", i - j);

    j = lps[j - 1];

    }

    else if (i < N && pat[j] != txt[i]) {

        if (j != 0)

        j = lps[j - 1];

        else

        i = i + 1;

        }

    }

}

void computeLPSArray(char*\** pat, int M, int*\** lps)

{

    int len = 0;

    lps[0] = 0;

    int i = 1;

    while (i < M) {

            if (pat[i] == pat[len]) {

        len++;

        lps[i] = len;

        i++;

        }

    else

        {

        if (len != 0) {

        len = lps[len - 1];

        }

        else

        {

        lps[i] = 0;

        i++;

        }

        }

    }

}

int main()

{

char txt[] = "ABABDABACDABABCABAB";

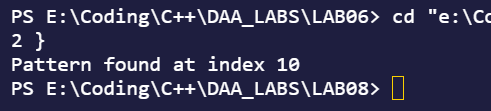
char pat[] = "ABABCABAB";

KMPSearch(pat, txt);

return 0;

}

**Output:**

****