**DAA LAB**

**L45-L46**

**EX-10**

**Name:** Preyash **Registration Number:** 20BPS1022

1. **Rabin Karp**

**Code:**

#include <bits/stdc++.h>

using namespace std;

#define d 256

void search(char pat[], char txt[], int q)

{

    int M = strlen(pat);

    int N = strlen(txt);

    int i, j;

    int p = 0;

    int t = 0;

    int h = 1;

    for (i = 0; i < M - 1; i++)

        h = (h \* d) % q;

    for (i = 0; i < M; i++)

    {

        p = (d \* p + pat[i]) % q;

        t = (d \* t + txt[i]) % q;

    }

    for (i = 0; i <= N - M; i++)

    {

        if ( p == t )

        {

            bool flag = true;

            for (j = 0; j < M; j++)

            {

                if (txt[i+j] != pat[j])

                {

                  flag = false;

                  break;

                }

            }

            if (j == M)

                cout<<"The given pattern is found at index: "<< i<<endl;

        }

        if ( i < N-M )

        {

            t = (d\*(t - txt[i]\*h) + txt[i+M])%q;

            if (t < 0)

            t = (t + q);

        }

    }

}

int main()

{

    char txt[] = "PRDITASHSAASERTWQPRDITASHSAPRDIQWERTY";

    char pat[] = "PRDI";

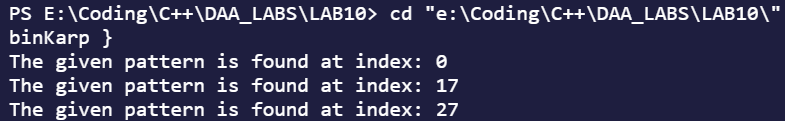
    int q = 101;

      search(pat, txt, q);

    return 0;

}

**Output:**

****

1. **Pattern matching using Finite Automata**

**Code:**

#include <bits/stdc++.h>

#include<stdio.h>

#include<string.h>

#define totalChar 256

int nextStateCalc(char *\**pat, int M, int state, int x) {

   if (state < M && x == pat[state])

      return state+1;

   int ns, i;

   for (ns = state; ns > 0; ns--) {

      if (pat[ns-1] == x) {

         for (i = 0; i < ns-1; i++)

            if (pat[i] != pat[state-ns+1+i])

               break;

         if (i == ns-1)

            return ns;

      }

   }

   return 0;

}

void TFcalc(char *\**pat, int M, int TF[][totalChar]) {

   int state, x;

   for (state = 0; state <= M; ++state)

      for (x = 0; x < totalChar; ++x)

         TF[state][x] = nextStateCalc(pat, M, state, x);

}

void occurences(char *\**pat, char *\**txt) {

   int M = strlen(pat);

   int N = strlen(txt);

   int TF[M+1][totalChar];

   TFcalc(pat, M, TF);

   int i, state=0;

   for (i = 0; i < N; i++){

      state = TF[state][txt[i]];

      if (state == M)

         printf ("The given pattern was found at the index: %d \n",i-M+1);

   }

}

int main() {

   char \*txt = "PRDITASHSAASERTWQPRDITASHSAPRDIQWERTY";

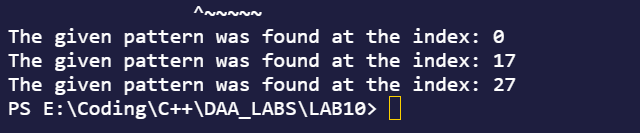
   char \*pat = "PRDI";

   occurences(pat, txt);

   return 0;

}

**Output:**

****

1. **Activity Selection problem using greedy method.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

#define N 6

struct activity

{

    int start, finish;

};

bool activitySort(activity s1, activity s2)

{

    return (s1.finish< s2.finish);

}

void maxActivityPrint(activity arr[], int n)

{

    sort(arr, arr+n, activitySort);

    cout<< "Following activities are selected \n";

    int i = 0;

    cout<< "(" <<arr[i].start<< ", " <<arr[i].finish << ")\n";

    for (int j = 1; j < n; j++)

    {

        if (arr[j].start>= arr[i].finish)

        {

            cout<< "(" <<arr[j].start<< ", "<<arr[j].finish << ") \n";

            i = j;

        }

    }

}

int main()

{

    activity arr[N];

    for(int i=0; i<=N-1; i++)

    {

        cout<<"Enter the start and end time of "<<i+1<<" activity \n";

        cin>>arr[i].start>>arr[i].finish;

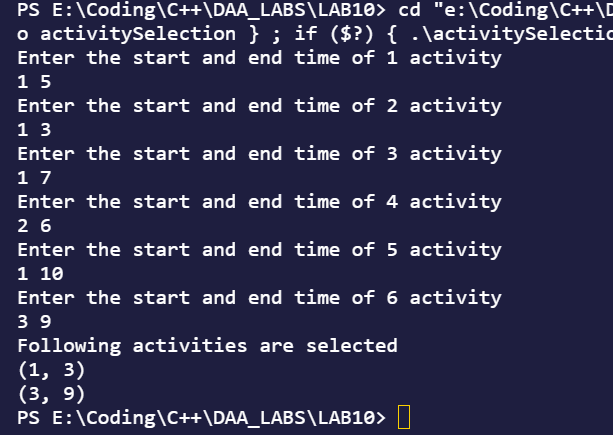
    }

    maxActivityPrint(arr, N);

    return 0;

}

**Output:**

****