/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

/\*\*

 \*

 \* @author hppc

 \*/

//apackage com.sanfoundry.setandstring;

import java.util.Scanner;

public class SearchStringUsingDFA {

    public static final int NO\_OF\_CHARS = 256;

    public static int getNextState(char[] pat, int M, int state, int x)

    {

        /\*

         \* If the character c is same as next character in pattern,

         \* then simply increment state

         \*/

        if (state < M && x == pat[state])

            return state + 1;

        int ns, i;

        /\*

         \* ns stores the result which is next state

         \* ns finally contains the longest prefix which is also suffix

         \* in "pat[0..state-1]c"

         \* Start from the largest possible value and stop when you find

         \* a prefix which is also suffix

         \*/

        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;

    }

    /\*

     \* This function builds the TF table which represents Finite Automata for a

     \* given pattern

     \*/

    public static void computeTF(char[] pat, int M, int[][] TF)

    {

        int state, x;

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

            for (x = 0; x < NO\_OF\_CHARS; ++x)

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

    }

    /\*

     \* Prints all occurrences of pat in txt

     \*/

    public static void search(char[] pat, char[] txt)

    {

        int M = pat.length;

        int N = txt.length;

        int[][] TF = new int[M + 1][NO\_OF\_CHARS];

        computeTF(pat, M, TF);

        // Process txt over FA.

        int i, state = 0;

        for (i = 0; i < N; i++)

        {

            state = TF[state][txt[i]];

            if (state == M)

            {

                System.out.print(pat);

                System.out.print(" found at " + (i - M + 1));

            }

            else

            {

                System.out.println("Not found");

            }

                }

    }

    public static void main(String[] args)

    {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the main string: ");

        String main = sc.nextLine();

        System.out.println("Enter the pattern string: ");

        String pattern = sc.nextLine();

        search(pattern.toCharArray(), main.toCharArray());

        sc.close();

    }

}