
Lab01:

1. Bruteforce Algorithm for string match

```
#include<iostream>
```

```
#include<string>
```

```
using namespace std;
```

```
int stringMatcher(string string_to_match , string sample_string)
```

```
{
```

```
    int len_string_to_match, len_sample_string;
```

```
    len_string_to_match = string_to_match.length();
```

```
    len_sample_string = sample_string.length();
```

```
    for(int s = 0 ; s < len_sample_string ; s++)
```

```
    {
```

```
        int i = 0, j = s;
```

```
        while(sample_string[j] == string_to_match[i])
```

```
        {  
            j++;  
            i++;  
  
            if (i == len_string_to_match)  
            {  
                cout << "string matched";  
                return 0;  
            }  
        }  
  
    }  
  
    return 0;  
}  
  
int main()  
{  
    string string_to_match;  
    string sample_string;
```

```
cout << "Enter the string to match: ";

cin >> string_to_match;

cout << "enter the sample string: ";

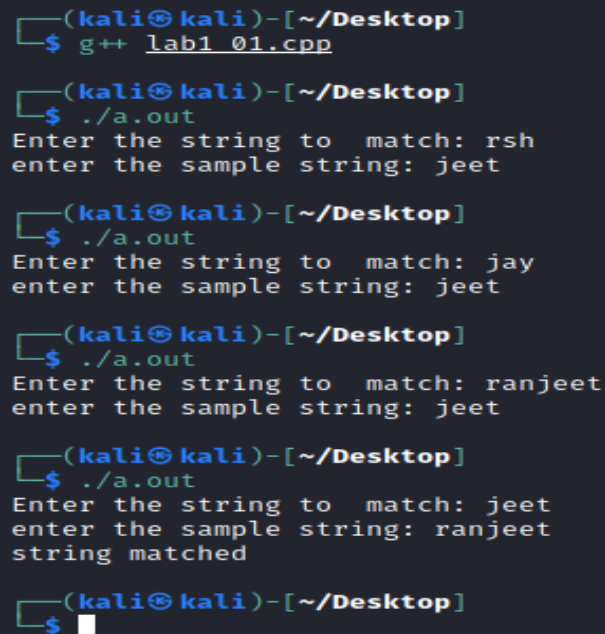
cin >> sample_string;

stringMatcher(string_to_match, sample_string);

return 0;

}
```

OUTPUT:



```
(kali㉿kali)-[~/Desktop]
$ g++ lab1_01.cpp

(kali㉿kali)-[~/Desktop]
$ ./a.out
Enter the string to match: rsh
enter the sample string: jeet

(kali㉿kali)-[~/Desktop]
$ ./a.out
Enter the string to match: jay
enter the sample string: jeet

(kali㉿kali)-[~/Desktop]
$ ./a.out
Enter the string to match: ranjeet
enter the sample string: jeet

(kali㉿kali)-[~/Desktop]
$ ./a.out
Enter the string to match: jeet
enter the sample string: ranjeet
string matched

(kali㉿kali)-[~/Desktop]
$
```

2. Horspool's Algorithm

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

using namespace std;

int stringMatcher(string sentence ,string pattern)
{
    int len_pattern ,len_sentence;
    len_pattern = pattern.length();
    len_sentence = sentence.length();
    int shift_table[27];
    for(int i = 0 ; i < 27 ; i++)
        shift_table[i] = len_pattern;

    int index;
    for(int j = 0 ; j < len_pattern - 1 ; j++)
    {
        index =  int(pattern[j]) - int('a');
        shift_table[index] = len_pattern - 1 - j;
    }
}
```

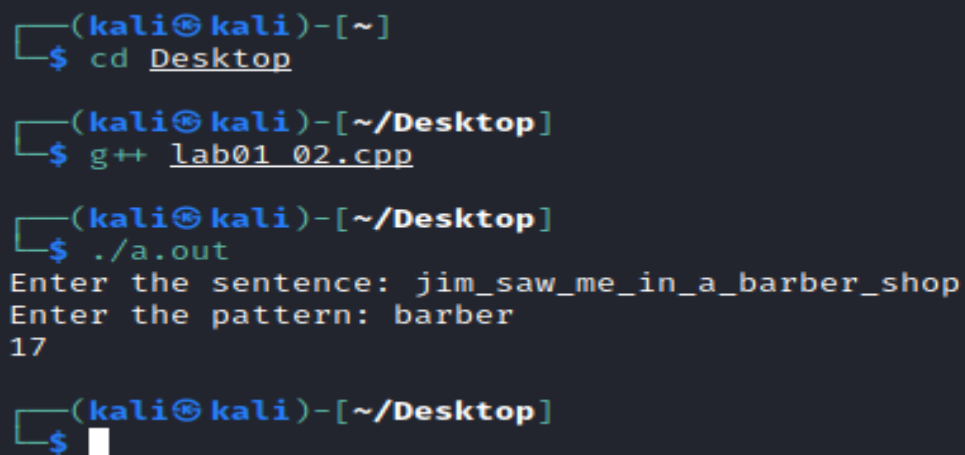
```

int k = 0;
while(k + len_pattern < len_sentence)
{
    if(pattern[len_pattern - 1] == sentence[len_pattern - 1 +
k])
    {
        int o;
        o = len_pattern - 2;
        while( o >= 0 && pattern[o] == sentence[o + k])
        {
            o = o - 1;
        }
        if( o == -1)
            return k + 1;
    }
    if(sentence[len_pattern - 1 + k] == '_')
        k = k + 6;
    else
        k = k + shift_table[int(sentence[len_pattern + k - 1])
- int('a')];
}
return len_sentence;
}

```

```
int main()
{
    string sentence, pattern;
    cout << "Enter the sentence: ";
    cin >> sentence;
    cout << "Enter the pattern: ";
    cin >> pattern;
    int result;
    result = stringMatcher (sentence, pattern);
    cout << result;
    return 0;
}
```

OUTPUT:



```
(kali㉿kali)-[~]
$ cd Desktop

(kali㉿kali)-[~/Desktop]
$ g++ lab01_02.cpp

(kali㉿kali)-[~/Desktop]
$ ./a.out
Enter the sentence: jim_saw_me_in_a_barber_shop
Enter the pattern: barber
17

(kali㉿kali)-[~/Desktop]
$
```