DAA

CSE2012

LAB-08

LPS

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

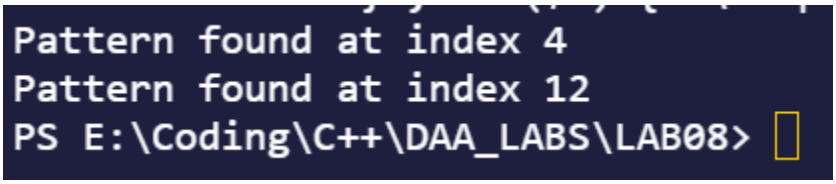
1.) **Optimized Naive approach for String matching**

Code:

```
#include <bits/stdc++.h>
using namespace std;
void search(string pat, string txt)
{
int M = pat.size();
int N = txt.size();
int i = 0;
while (i <= N - M)
{
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;
i = i + M;
}
else if (j == 0)
i = i + 1;
else
i = i + j;
}
}
int main()
{
string txt = "ABCEABCDABCEABCD";
string pat = "ABCD";
search(pat, txt);
```

```
return 0;
}
```

Output:



2.) The user enters a character in place of the dollar sign and then perform string pattern searching using optimized naïve approach.

Code:

```
#include <bits/stdc++.h>
using namespace std;
void search(string pat, string txt)
{
int M = pat.size();
int N = txt.size();
int i = 0;
while (i <= N - M)
{
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;
i = i + M;
}
else if (j == 0)
i = i + 1;
else
i = i + j; // slide the pattern by j
}
}
/* Driver code*/
int main()
{
string txt = "ABCEABCDABCEABCD";
cout<<"Enter the value of dollar AB$D";
```

```
string k;
cin>>k;
string pat = "AB"+k+"D";
search(pat, txt);
return 0;
}
```

Output:

```
Enter the value of dollar AB$DC
Pattern found at index 4
Pattern found at index 12
PS E:\Coding\C++\DAA_LABS\LAB08>
```

3.) String pattern searching where dollar can be any character or characters.

Code:

```
#include <bits/stdc++.h>
using namespace std;
int search(string pat, string txt)
{
 int M = pat.size();
 int N = txt.size();
 int i = 0;
 while (i <= N - M)
 {
 int j;
 for (j = 0; j < M; j++)
 if (txt[i + j] != pat[j])
 break;
 if (j == M)
 {
 return i;
 i = i + M;
 }
```

```
 else if (j == 0)
 i = i + 1;
 else
 i = i + j; // slide the pattern by j
 }
 return 0;
}
int main()
{
 string txt = "ABCEABCDABCEABCD";
 cout << "check for regex BC$ABCD where $ can be any character" << endl;
 string pat1 = "BC";
 string pat2 = "ABCD";
 int i1 = search(pat1, txt);
  int i2 = search(pat2, txt)+ pat2.length()-1;
 if (i1 < i2)
 {
 cout << "Pattern found at index " << i1 << " to " << i2;
 }
 return 0;
}
```

Output:

```
Pattern found at index 10 check for regex BC$ABCD where $ can be any character
Pattern found at index 1 to 7
```