CET 214 – Data Structures & Algorithms

Experiment # 6

Experiment Title

String Processing and Pattern Matching

Assessment of CLO(s): III

Performed on 25-10-2024

Student Name			
Roll No.		**Group**	
Semester		**Session**	

Total (Max)	Criteria 1 (2.5)	Criteria 2 (2.5)	Criteria 3 (2.5)	Criteria 4 (2.5)	Total (10)
Marks Obtained					
Remarks (if any)					

Experiment evaluated by

Instructor's Name	**Engr. Muhammad Asad Husain**		
Date		**Signature**	

Department of Engineering Technology
(UIT University)

Course Code: CET214 **Course Title: Data Structures & Algorithms** **Course Credits: 2+1** **Session: Fall 2024**

Rubric for assessment criteria to perform experiment number 6.

Level / Criteria	UNSATISFACTORY 1	COMPETENT 2	PROFICIENT 3	DISTINGUISHED 4
Capability of writing algorithm/ Procedure	None of the steps are implemented of an algorithm.	Few steps are implemented correctly of an algorithm.	Most of the steps are implemented correctly of an algorithm.	All the steps are implemented correctly of an algorithm.
Capability of writing Program	Programs not completed.	Completeness of code, consistent variable naming and unformatted.	Completeness of code, inconsistent variable naming and well formatted.	Completeness of code, consistent variable naming and well formatted.
Completion of target in Lab	25% target has been completed	50% target has been completed	75% target has been completed	100% target has been completed
Output	None of the outputs are correct.	Few outputs have been found correctly.	Some of the outputs are correct and well formatted.	Most of the outputs are correct and well formatted.

Practical Objective(s):

1. Getting familiar with basic string operations
2. Learning to use various built-in functions for performing basic string operations
3. Implementing insertion, deletion and replacement operations on strings

Theory

String Processing:
A finite sequence S of zero or more characters is called string. The number of characters in a string is called its length. The string with zero characters is called empty string or null string.

Below are given some basic string operations:
- **Substring**: This operation extracts a subset of the given string.
- **Indexing**: Indexing refers to finding the position where a string pattern first appears in the given string text.
- **Length**: The number of characters in a string is called its length.
- **Concatenation**: Let S1 and S2 be two strings. Concatenation of S1 and S2 denoted by S1||S2 is the string consisting of the characters of S1 followed by S2.

In earlier times, character data processed by the computer consisted mainly of data items, such as names and addresses. Today the computer also processes printed matter, such as letters, articles and reports. It is in the latter context that we use the term ''word processing".

The operations usually associated with word processing are the following:
- **Insertion**: Inserting a string in the middle of the text
- **Deletion**: Deleting a string from a text
- **Replacement**: Replacing one string in the text by another.

Pattern Matching:
Pattern matching is the problem of deciding whether the given pattern string pattern P appears in a text T. We are assuming that the length of pattern P does not exceed the length of text T.

String Processing:

Example 1: Performing various operations on a string

```
#include <iostream>
#include <string> using
namespace std; int main ()
{
    string str1 (" This sentence is now");
    string str2 (" complete ");

    size_t found_index = str1.find('s');      //indexing
    string sub_str = str1.substr(6,8);        //substring
    size_t len1 = str1.length();              // length string
    con_str = str1+str2;                      //concatenation

    cout<<"substring="<<sub_str<<endl;
    cout<<"index of first found pattern="<<found_index<<endl;

    cout<<"length of string="<<len1<<endl; cout<<"concatenated
    string="<<con_str<<endl;
    return 0;
}
```

Algorithm 1: This algorithm inserts a string S in a given text T at position K.

```
INSERT (text, position, string)

Step 1:   [Initialize] Set T: =text, K: =position and S: =string
Step 2:   Set S1 = SUBSTRING (T, 1, K-1)
Step 3:   Set S2 = SUBSTRING (T, K, LENGTH (T)-K+1).
Step 4:   [Concatenate] Set S3 = S1||S||S2
Step 5:   Write: S3
Step 6:   Exit
```

Code 1:

```
#include <iostream>
#include <string>
using namespace std;

string insert(string text, int position, string add_text)
{
    size_t len1 = text.length();
    string str1 = text.substr(1,position-1);
    string str2 = text.substr(position,len1-position+1);
    string str3 = str1+add_text+str2;
    cout << "Updated sentence="<<str3<<endl;
    return str3;
}
```

```
int main ()
{
    string text1= (" Programming is fun! ");
    string add_text= ("not ");
    size_t position = 16;
    insert(text1, position, add_text);
    return 0;
}
```

Algorithm 2: This algorithm deletes a string of length L from a given text T. K is the position of the first character to be deleted.

DELETE (text, position, length)

```
Step 1:    [Initialize] Set T = text, K = position and L = length
Step 2:    Set S1 = SUBSTRING (T, 1, K-1)
Step 3:    Set S2 = SUBSTRING (T, K+L, LENGTH (T)-K-L+1)
Step 4:    [Concatenate] Set S3: =S1||S2
Step 5:    Write: S3
Step 6:    Exit
```

Algorithm 3: This algorithm replaces a pattern P1 with another pattern P2 in a given text T.

REPLACE (text, pattern1, pattern2)

```
Step 1:    [Initialize] Set T: =text, P1: =pattern1 and P2: =pattern2
Step 2:    K = location of first occurrence of P1 in T.
Step 3:    T = DELETE (T, K, LENGTH(P1))
Step 4:    S = INSERT (T, K, P2)
Step 5:    Write: S
Step 6:    Exit
```

Pattern Matching Algorithm:

Algorithm 4: P is the string pattern which is to be matched in the given string T. The length of string T is S. The length of string pattern P is R. This algorithm will find the index of P in T.

```
Step 1:    [Initialize] set K: =0 and MAX: =S-R+1

Step 2:    Repeat Steps 3 to 5 while K<MAX

Step 3:    Repeat for L=0 to R: [Test each character of P]
           If P [L] is not equal to T [K+L], then: Go to step 5.
           [End of step 3 loop (inner loop)]

Step 4:    [Success] Set INDEX = K and Exit

Step 5:    Set K: =K+1
           [End of Step2 loop (outer loop)]

Step 6:    [Failure] Set INDEX = -1

Step 7:    Exit
```

Code 2:

```
#include <iostream>
#include <library>
using namespace std;

int main ()
{
    string text = ("Find x");
    string pattern = ("x");
    //declare INDEX
    //declare and initialize R
    //declare and initialize S
    //declare MAX
    //declare and initialize K
    //declare L

    while (condition)
    {
         for (L=0;condition;L++)
         {
              if (condition)
                   break;
```

```
        }
              if(L==R)
              {
                    INDEX=K;
                    break;
              }
              else
                    K=K+1;

  }

  if(K>MAX)
        statement
  if (INDEX!=-1)
        cout<<"Index of "<<pattern<<" in "<<text<<" is "<<INDEX<<endl;
  else
        cout<<"Pattern not found in given string";
        system("pause");
  return 0;
  }
```

Do It Yourself:

1. Implement algorithm 2 in C++.
2. Implement algorithm 3 in C++.
3. Complete the pseudo code given in **Code 2** and observe the output.
4. Execute the program you have completed in 3. Use the below mentioned strings and write the INDEX for each pair:

    a. text = **baaanabanana**, pattern = **<u>banana</u>**
    b. text = **whatareyoudoing**, pattern = **<u>you</u>**
    c. text = **wethepeople**, pattern = **<u>Booo</u>**