QUOTATIONS PROBLEM USING RECURSION IN C++

In this programming assignment, you will practice recursion.

> *"All the world's a stage, and all the men and women merely players."*
> *William Shakespeare*

When we write, we use single quotation or double quotation marks to mark a quote. For a single quote, a pair of single quotation marks (') is used. For a double quote, a pair of double quotation marks (") is used. For processing textual documents (strings), we may need to match the quotation marks to find a quote. An sample application could be turnitin. Turnitin can exclude quotation marks when comparing documents.

In this assignment, we will be implementing recursive functions to process quotation marks, such as finding a quotation mark, matching quotation marks, etc.

Tasks

You have to complete five tasks in this programming assignment.
Since this is an assignment on recursion, you are *NOT* allowed to use any kind of loops.
In addition, you are *not* allowed to use global variables or include any additional libraries.
Othersie, you will receive ZERO marks for the assignment.
You can add additional helper function(s) if necessary.

All following functions have 2 formal parameters:

- line: specifies the input C String, which is a NULL-terminated char array.
- start: specifies the starting index. It is always 0 when we call the function in main().

Practice Task (Not Graded)

Task 0 (0 points)

```
unsigned int recursive_strlen(const char line[], int start)
```

- This function implements the strlen() function recursively.
- Return: Number of characters in the string (line), *excluding* the NULL character at the end.

Hints

- In the lecture notes of "Recursion", str_len() is implemented using a loop. You can use it for reference in thinking about how to convert the loop into recursion.
- How can you process one char at each recursive call?
- Where does the loop end? At which condition should the recursion end?
- How does the loop move to the next character? How can you do the same using recursion?

Sample Input (line)	Return Value
(Empty)	0
A	1
COMP2011	8

Normal Tasks

Task 1 (15 points)

`unsigned int count_dquotes(const char line[], int start)`

- This function counts the number of double quotation characters (") in the string (`line`).
- Return: Number of double quotation characters.

Hint

- This should be similar to Task 0.

Sample Input (line)	Return Value
(Empty)	0
COMP2011	0
"COMP2011"	2

Task 2 (15 points)

`int find_first_dquote(const char line[], int start)`

- This function finds the index of the first double quotation character in the string (`line`).
- Return:
  - `ERROR` (see the global constants in `pa2.cpp`) if no double quotation characters are found.
  - Index of the first double quotation character if otherwise.

Hints

- Start by considering the case that double quotation characters are always found.
- Then move on to consider the case when no double quotation characters are found.
- What does it mean by reaching the end of string? What should you return?
- How should you pass back this information?

Sample Input (line)	Return Value
(Empty)	ERROR
COMP2011	ERROR

Sample Input (line)	Return Value
"COMP2011"	0
Hello " World	6

Task 3 (20 points)

`int count_chars_in_matched_dquote(const char line[], int start)`

- This function counts the number of characters inside all pairs of matching double quotation characters in the string (`line`).
- Return:
  - `ERROR` if some double quotation characters are not matched.
  - Number of characters inside all pairs of matched double quotation characters if otherwise.

Definition of Matching Double Quotation Characters

- Starting from the beginning of the string, a double quotation character is matched with the next closest double quotation character.
- For example, consider the following string:

  `"Hello World", "COMP 2011"`

  - The double quotation character before `Hello` is matched with the double quotation character after `World`.
  - The double quotation character before `COMP` is matched with the double quotation character after `2011`.
  - All double quotation chacters are matched in this string. Returns 20.
- Another example:

  `"Matched" Some"Quote`

  - The double quotation chacters around `Matched` are matched.
  - But the double quotation characters between `Some` and `Quote` is not matched with any double quotation chacters. Returns `ERROR`.

Hint

- What does it mean when a double quotation character is read?
- Write a helper function.

Sample Input (line)	Return Value
(Empty)	0
COMP2011	0
"COMP2011"	8
Hello " World	ERROR

Sample Input (line)	Return Value
"Hello" "World"	10
" "Hello World" "	2
"A double quote: ""	ERROR

Challenging Tasks

Task 4 (25 points)

`bool check_quotes_matched(const char line[], int start)`

- This function checks whether quotation characters (single and double quotation characters) in the string (`line`) are matched.
- Return:
  - `true` if all quotation characters are matched, or no quotation characters are found.
  - `false` otherwise.

Definition of Matching Quotation Characters

- Starting from the beginning of the string,
  - A single quotation character is matched with the next closest single quotation character.
  - A double quotation character is matched with the next closest double quotation character.
- Exceptions:
  - Single quotation characters inside a pair of matched double quotation characters are ignored.
  - Double quotation characters inside a pair of matched single quotation characters are ignored.
- Examples:

  `"Hello World", 'COMP 2011'`

  - Double quotation characters around `Hello World` are matched.
  - Single quotation characters around `COMP 2011` are matched.
  - Returns `true`.

  `"Hello 'World', '" "`

  - Double quotation characters around the whole string are matched.
  - All single quotation characters are inside a pair of double quotation characters, thus they are ignored.
  - All quotation characters are matched. Returns `true`.

  `'Hello"COMP'2011"`

    - The single quotation character before `Hello` is matched with the single quotation character after `COMP`.
    - The double quotation character between `Hello` and `COMP` is ignored.
    - The double quotation character after `2011` is not matched with any double quotation characters.
    - Returns `false`.

Hint

- Write two helper functions to handle:
    1. When a single quotation character is read.
    2. When a double quotation character is read.
- These three functions should call each other.

Sample Input (line)	Return Value
(Empty)	true
COMP2011	true
'COMP2011'	true
Hello " World	false
" 'Hello World' "	true
"A single quote: '"	true
"A'A'A'A"	true
'"A""A"	false

Task 5 (25 points)

```
unsigned int length_of_longest_consecutive_dquotes(const char line[], int start)
```

- This function finds the length of the longest consecutive sequence of double quotation characters in the string (`line`).
- It returns the length of the longest consecutive sequence of double quotation characters.

Hint

- Write a recursive helper function that takes additional parameters.

Sample Input (line)	Return Value
(Empty)	0
COMP2011	0
"COMP2011"	1
Hello " World	1
"""Hello World"""	3
AAAAAA""BBB	2

Sample Input (line)	Return Value
"""AAA"""""""	6
"""""BBBBBBBBB"""	5

SOLUTION

```
/*
 * Note:
 * - DO NOT change any of the function headers given
 * - DO NOT use any loops
 * - DO NOT use any global variables or add additional libraries.
 * - You can add helper function(s) if needed.
 */

#include <iostream>

using namespace std;

// Constants

// NULL character. This is the last char of all C Strings
const char END = '\0';

// Single quotation character '
const char SQUOTE = '\'';

// Double quotation character "
const char DQUOTE = '\"';

// Error. Used in Task 2 and 3
const int ERROR = -1;

// Practice Task: Task 0 (Not Graded)
unsigned int recursive_strlen(const char line[], int start)
{
   if (line[start] != END){
      return 1 + recursive_strlen(line, start+1);
   } else {
      return 0;
   }
}


// Normal Task: Task 1
unsigned int count_dquotes(const char line[], int start)
{
   if (line[start] != END){
      if (line[start] == DQUOTE){
         return 1 + count_dquotes(line, start+1);
```

```
    } else {
      return count_dquotes(line, start+1);
    }
  } else {
    return 0;
  }
}


// Normal Task: Task 2
int find_first_dquote(const char line[], int start)
{
  if (line[start] != END){
    if (line[start] == DQUOTE){
      return start;
    } else {
      return find_first_dquote(line, start+1);
    }
  } else {
    return ERROR;
  }
}


// Normal Task: Task 3
int count_chars_in_matched_dquote(const char line[], int start)
{
  if (line[start] != END){
    if (line[start] == DQUOTE){
      int next_start = find_first_dquote(line, start+1);

      if (next_start == ERROR){
        return ERROR;
      } else {
        int count = count_chars_in_matched_dquote(line, next_start+1);

        if (count == ERROR){
          return ERROR;
        } else{
          return (next_start - start - 1 + count);
        }
      }
    } else {
      return count_chars_in_matched_dquote(line, start+1);
    }
  } else {
    return 0;
  }
}

// helper function for finding first single quote
```

```
int find_first_squote(const char line[], int start)
{
   if (line[start] != END){
      if (line[start] == SQUOTE){
         return start;
      } else {
         return find_first_squote(line, start+1);
      }
   } else {
      return ERROR;
   }
}


// Challenging Task: Task 4
bool check_quotes_matched(const char line[], int start)
{
   if (line[start] != END){
      if (line[start] == SQUOTE){
         int next_start = find_first_squote(line,start+1);

         if (next_start == ERROR){
            return false;
         } else {
            return check_quotes_matched(line,next_start+1);
         }

      } else
         if (line[start] == DQUOTE){
            int next_start = find_first_dquote(line,start+1);

            if (next_start == ERROR){
               return false;
            } else {
               return check_quotes_matched(line,next_start+1);
            }
         } else {
            return check_quotes_matched(line, start+1);
         }


   } else {
      return true;
   }
}

int helper_consequtive_dquotes(const char line[], int start, int count)
{
   if (line[start] == DQUOTE ){
      return helper_consequtive_dquotes(line, start+1, count+1);
   } else {
```

```
        return count;
    }
}


// Challenging Task: Task 5
unsigned int length_of_longest_consecutive_dquotes(const char line[], int start)
{
    if (line[start] != END){

        //cout << " Character : " << line[start] << endl;
        int cnt_dquotes = 0;
        if (line[start] == DQUOTE){
            cnt_dquotes = helper_consequtive_dquotes(line,start+1,1);
            int len_dquotes = length_of_longest_consecutive_dquotes(line,start+cnt_dquotes+1);

            //cout << "Temporary cnt_dquotes  : " << cnt_dquotes << "  , len_dquotes : " <<
len_dquotes << endl;
            if (cnt_dquotes >= len_dquotes){
                return cnt_dquotes;
            } else {
                return len_dquotes;
            }
        } else {
            return length_of_longest_consecutive_dquotes(line, start+1);
        }
    }
}


// DO NOT WRITE ANYTHING AFTER THIS LINE. ANYTHING AFTER THIS LINE WILL BE
REPLACED

const int MAX_LENGTH = 1000;

int main()
{
    int option = 0;
    char line[MAX_LENGTH];

    do {
        cout << "Options:" << endl;
        cout << "0:  Test recursive_strlen()" << endl;
        cout << "1:  Test count_dquotes()" << endl;
        cout << "2:  Test find_first_dquote()" << endl;
        cout << "3:  Test count_chars_in_matched_dquote()" << endl;
        cout << "4:  Test check_quotes_matched()" << endl;
        cout << "5:  Test length_of_longest_consecutive_dquotes()" << endl;
        cout << "-1: Quit" << endl;

        cin >> option;
```

```
cin.ignore();

switch (option) {
   case 0:
   cout << "Testing recursive_strlen()" << endl;
   cout << "Enter line: ";
   cin.getline(line, MAX_LENGTH);
   cout << recursive_strlen(line, 0) << endl;
   break;

   case 1:
   cout << "Testing count_dquotes()" << endl;
   cout << "Enter line: ";
   cin.getline(line, MAX_LENGTH);
   cout << count_dquotes(line, 0) << endl;
   break;

   case 2:
   cout << "Testing find_first_dquote()" << endl;
   cout << "Enter line: ";
   cin.getline(line, MAX_LENGTH);
   cout << find_first_dquote(line, 0) << endl;
   break;

   case 3:
   cout << "Testing count_chars_in_matched_dquote()" << endl;
   cout << "Enter line: ";
   cin.getline(line, MAX_LENGTH);
   cout << count_chars_in_matched_dquote(line, 0) << endl;
   break;

   case 4:
   cout << "Testing check_quotes_matched()" << endl;
   cout << "Enter line: ";
   cin.getline(line, MAX_LENGTH);
   cout << check_quotes_matched(line, 0) << endl;
   break;

   case 5:
   cout << "Testing length_of_longest_consecutive_dquotes()" << endl;
   cout << "Enter line: ";
   cin.getline(line, MAX_LENGTH);
   cout << length_of_longest_consecutive_dquotes(line, 0) << endl;
   break;

   default:
   break;
}

cout << endl;
```

```
    } while (option != -1);

    return 0;
}
```