Lab 11 - Recursion

Due: end of class

OVERVIEW

For this lab, you will get some practice writing recursive functions and understanding recurrence relations.

Task 0:

There are many types of bracket symbols: {}, (), [], and <>. A string is set to be well-formed if the symbols are properly nested. For example:

Well-Formed	Not Well-Formed
{[{}]}	{(){
<[()]>	<[()]()>
<<(<()>)>>	{(})

For this task, we wish to determine whether a string is well-formed. This is easy to express recursively - A string s composed solely of bracketing symbols is well-formed if:

1. Its first and last characters are a matching pair
2. The rest of the string is also well-formed (this is the recursive assumption).

Formally, this can be expressed as the following recurrence relation where n is the length of s:

wellFormed(x) = match(x[0],x[n-1]) AND wellFormed(x[1..n-2])

where match(c1,c2) is true if c1 is a left bracketing symbol and c2 is the corresponding right one. Don't forget to add the base case/s.

Write a program that inputs a string from the user, calls wellFormed, and prints a message indicating whether or not the string is well-formed. You may use the string library function `substr` to extract the substring in the recurrence relation.

Task 1:

Again, consider strings of bracket symbols. Now we wish to determine the level of nesting. For example, the levels for the 3 well-formed strings in the above table are 3, 3, and 5 respectively. Write a recursive function that outputs the nesting level of an argument string using the following prototype:

```
int nestLevel(string s);
```

State your recurrence relation in comments. You may assume the precondition that s is well-formed.

Task 2:

Task 0 is also useful in parsing programs (and English sentences and math expressions). However programs also contain characters other than brackets. One way of doing this is to first strip the string of non-bracket symbols before calling wellFormed. But in this task, you should modify the recursive function wellFormed instead.

HAND IN

Your 136 instructor will tell you what to hand in and how.

GENERAL COMMENTS FOR ALL PROGRAMS THIS SEMESTER

You should have the following header on all programs:

```
/*
  Author: <name>
  Course: {135,136}
  Instructor: <name>
  Assignment: <title, e.g., "Lab 1">

  This program does ...
*/
```

GRADING

All 135 and 136 programs this semester will be graded on:

- Correctness: Does your program work?
- Testing: Have you generated sufficient and good test data to give reasonable confidence that your program works?
- Structure: Have you structured your code to follow proper software engineering guidelines? This includes readability and maintainability.
- Documentation: How well documented is your code? Good documentation does not repeat the code in English, but explains the point of each code block, highlighting any design decisions and/or tricky implementation details.