Version 1.0 ***due:*** *Monday, September 9 @ 11:59pm*

1 Introduction

This assignment is meant to help you familiarize yourself with the hand-in mechanism for 15-210 and to get you thinking about proofs and coding again. To that end, you will answer some questions about the mechanics and infrastructure of the course, implement two solutions to the parenthesis distance problem, and perform some analysis of your solutions. Finally, you will complete some exercises involving big-O.

2 Files

After downloading the assignment tarball from Autolab, extract the files from it by running `tar -xvf parenlab-handout.tgz` from a terminal window. You should see the following files:

1. `parenlab.pdf`
2. `bobbfile.py`
3. `Makefile`
4. `sources.cm`
5. `support.cm`
6. `lib/`
7. `PAREN.sig`
8. `ArrayParenPackage.sml`
9. `Tester.sml`
10. `MkSequentialPD.sml`
11. * `MkBruteForcePD.sml`
12. * `MkDivideAndConquerPD.sml`
13. * `Tests.sml`

You should only modify the last 3 files, denoted by *. Additionally, you should create a file called `written.pdf` which contains the answers to the written part of the assignment.

3 Submission

Before submitting, ensure that `written.pdf` exists and is a valid PDF document. Then run `./submit` from within the `parenlab` directory.

Important note: You will still need to visit the Autolab website to view the results of the tests that Autolab runs against your code (which will contribute to part of your grade for this assignment). It is important to always do this for every lab to make sure that your code runs as you expect it to.

4 Mechanics

The following questions are intended to make sure that you have read and understood the various policies (see `http://cs.cmu.edu/~15210/policy.html`) for the course, as well as found the tools that we've set up to communicate with you.

Task 4.1 (1%). Describe the picture posted as an instructor's note on Piazza. Be creative!

Task 4.2 (3%). In each of the following situations, have the students followed or broken the collaboration policy? Briefly justify your answers with a sentence or two.

1. Ludmilla, Joaquin, and Alexander have lunch together after 210 lecture. Over lunch, they discuss the homework assignment released earlier in the week, including their progress so far. After lunch, all three go to different classes and don't think about the 210 homework until that evening.

2. While working on 213 homework near his friends from 210, Jon has a moment of insight into the 210 homework assignment. He becomes excited, and tells his friends what he thought of. Ishmael hasn't gotten that far in the homework, so he doesn't quite understand what Jon is talking about. Nonetheless, Ishmael copies down what he thinks are the key bits of what he heard to look at when he gets that far.

3. Yelena has been working on the 210 homework but can't figure out why her solution isn't compiling. She asks Abida to work through a couple of toy examples of functor syntax together, and they do so with a text editor and the SML REPL. Afterwards, Yelena gets her homework to compile and keeps working towards a solution.

Task 4.3 (1%). If you hand in your homework 25 hours after the posted due date, and you have no late days remaining, what is your maxiumum possible score?

5 The Parenthesis Distance Problem

A string s 'closed' if and only if it contains only '(' and ')' characters and is one of the following:

- The concatenation of two closed strings, $s_1 s_2$.
- A single closed string s_0 surrounded by a pair of 'matched' parentheses, (s_0).
- A single pair of matched parentheses, ().

The distance between a pair of matched parentheses is the number of characters between the parentheses (exclusive). The maximum parenthesis distance problem is to find the largest distance between a pair of matched parentheses in a closed string. More formally, for the closed string s, let M_s be the set of index pairs such that for $(i, j) \in M_s$, $i < j$ and (s_i, s_j) is a pair of matched parentheses. The maximum parenthesis distance is

$$\max\{j - i - 1 | (i, j) \in M_s\}$$

For example, the string '(()())()', has a maximum parenthesis distance of 4.

5.1 Representation

When solving this problem, instead of interacting with strings, you will work with `paren` sequences, where the type `paren` is defined in a structure that ascribes to `PAREN_PACKAGE` and is given by

```
datatype paren = OPAREN | CPAREN
```

with `OPAREN` corresponding to a left parenthesis and `CPAREN` corresponding to a right parenthesis.

5.2 Implementation

You will implement two solutions of the function `parenDist : paren seq -> int option` such that `parenDist` $S \Rightarrow$ `NONE` if S is not closed and `parenDist` $S \Rightarrow$ `SOME` max if S is closed, where max is the maximum parenthesis distance in S.

Each solution will be a functor ascribing to the signature `PAREN_DIST`, defined in `PAREN.sig`. Each functor will take a structure ascribing to `PAREN_PACKAGE` as its only argument. The signature `PAREN` is defined in terms of the `SEQUENCE` signature from our library. For testing, you should use the structure `ArrayParenPackage`, which uses the `ArraySequence` implementation of `SEQUENCE`.

How to indicate parallelism? As seen in recitation, you should use the function `par` (inside the structure `Primitives`) to express parallel evaluation. Parallel operations can also be expressed in terms of operations on sequences such as `map` or `reduce`. *You must be explicit about what calls are being made in parallel to receive full credit.*

5.3 Brute-Force Algorithm

It is possible to give a brute-force algorithm by generating all possible solutions and picking the best. Note that this is different from the sequential solution which is provided for you in `MkSequentialPD.sml`.

Task 5.1 (15%). Complete the functor `MkBruteForcePD` in the file `MkBruteForcePD.sml` with a brute-force solution to the maximum parenthesis distance problem. You may use the the solution to the parenthesis matching problem from recitation 1. You may also find `Seq.subseq` to be useful for your solution. Please ensure that you understand the definition of a *brute-force* solution before attempting this task (we have received many non-brute-force solutions in the past).

Task 5.2 (5%). What is the work and span of your brute-force solution? You should assume `subseq` has $O(1)$ work and span, where m is the length of the resulting subsequence, and `parenMatch` has $O(n)$ work and $O(\log^2 n)$ span where n is the length of the sequence.

5.4 Divide-and-Conquer Algorithm

You will now implement a solution to the maximum parenthesis distance problem using divide-and-conquer recursive programming. The work and span of your solution must be expressed by the recurrences

$$W_{parenDist}(n) = 2 \cdot W_{parenDist}\left(\frac{n}{2}\right) + W_{showt}(n) + O(1)$$

$$S_{parenDist}(n) = S_{parenDist}\left(\frac{n}{2}\right) + S_{showt}(n) + O(1)$$

where n is the length of the input parenthesis sequence, and W_{showt} and S_{showt} are the work and span of `showt`, respectively. Assume that the work and span is constant for an input sequence of length 1. A solution with correct behavior but with work or span that is not described by the appropriate recurrence will not receive full credit.

Task 5.3 (30%). Complete the functor `MkDivideAndConquerPD` in the corresponding file with a divide-and-conquer solution as described above. For this assignment, you are not required to submit a proof of correctness of your implementation. However, we advise that you work out a proof by mathematical induction for your solution as an exercise.

Task 5.4 (20%). The specification in Task 5.3 stated that the work of your solution must follow a recurrence that was parametric in the work it takes to view a sequence as a tree. Naturally, this depends on the implementation of `SEQUENCE`.

1. Solve the work recurrence with the assumption that $W_{showt} \in \Theta(\lg n)$ where n is the length of the input sequence.
2. Solve the work recurrence with the assumption that $W_{showt} \in \Theta(n)$ where n is the length of the input sequence.

3. In two or three sentences, describe a data structure to implement the sequence α `seq` that allows `showt` to have $\Theta(\lg n)$ work.

4. In two or three sentences, describe a data structure to implement the sequence α `seq` that allows `showt` to have $\Theta(n)$ work.

5.5 Testing Your Code

Task 5.5 (5%). In this course you will be expected to test your code extensively. For this assignment you should make sure you thoroughly and carefully test both of your implementations of the `PAREN_DIST` signature. Your tests should include both edge cases and more general test cases on specific sequences.

To aid with testing we have provided a testing structure, `Tester`, which should simplify the testing process. `Tester` will test your implementations against test cases specified within `Tests.sml`. Test cases should be added as strings to the `tests` list in the `Tests` structure. Each test case string should consist of the characters '(' and ')' only, which `Tester` will translate into a `paren seq`.

In order to test your code, after running `CM.make`, you will need to call the following functions to test your brute-force and divide-and-conquer implementations respectively:

```
Tester.testBF ()
Tester.testDC ()
```

Only put your test cases in `Tests.sml`. There should be no testing code in any other file when you submit your solution. Unlike the 15-150 testing methodology, our testing structure does not test your code at compile time.

6 Asymptotics

For this problem, let's recall the definition of big-O:

Definition 6.1. A function $f : \mathbb{N} \to \mathbb{R}_+$ is in $O(g)$ if and only if there exist constants $N_0 \in \mathbb{N}$ and $c \in \mathbb{R}_+$ such that for all $n \geq N_0$, $f(n) \leq c \cdot g(n)$.

Task 6.1 (5%). Rearrange the list of functions below so that it is ordered with respect to O—that is, for every index i, all of the functions with index less than i are in big-O of the function at index i. You can just state the ordering; you don't need to prove anything.

1. $f(n) = n^{\lg(n^2)}$
2. $f(n) = 2n^{1.5}$ 7>3>1>4>6>2>5
3. $f(n) = (n^n)!$
4. $f(n) = 43^n$

5. $f(n) = \lg(\lg(\lg(\lg(n))))$
6. $f(n) = 36n^{52} + 15n^{18} + n^2$
7. $f(n) = n^{n!}$

Task 6.2 (15%). Carefully **prove** each of the following statements, or provide a counterexample and **prove** that it is in fact a counterexample. You should refer to the definition of big-O. Remember that verbose proofs are not necessarily careful proofs.

1. O is a transitive relation on functions. That is to say, for any functions f, g, h, if $f \in O(g)$ and $g \in O(h)$, then $f \in O(h)$.
2. O is a symmetric relation on functions. That is to say, for any functions f and g, if $f \in O(g)$, then $g \in O(f)$.

1.传递关系 2.对称关系 3.反对称关系

3. O is an anti-symmetric relation on functions. That is to say, for any functions f and g, if $f \in O(g)$ and $g \in O(f)$, then $f = g$.