ARIZONA STATE UNIVERSITY

CSE 310, All Sections — **Data Structures and Algorithms** — Fall 2015

Assignment #1

Available Monday, 08/24/2015; due electronically Friday, 09/11/2015

This assignment covers Chapters 2, 3, and 4 of the text [100 marks total].

Submit electronically, before midnight on Friday, 09/11/2015 using the submission link on Blackboard for Assignment #1, a file named `yourFirstName-yourLastName.pdf` containing your solution to this assignment (a .doc or .docx file is also acceptable, but .pdf is preferred).

1. Suppose the functions $g_1, \ldots, g_7$ are defined as follows: $g_1(n) = \sqrt{n}$, $g_2(n) = 2^n$, $g_3(n) = n^{1/3}$, $g_4(n) = (\log n)^2$, $g_5(n) = n^{\log n}$, $g_6(n) = n/\log n$ and $g_7(n) = (1/3)^n$. Arrange these seven functions in ascending order by their growth rate. In other words, if you place $g(n)$ after $f(n)$ then show that $f(n) = O(g(n))$. Fully justify your answer! [25 marks]

2. Consider the following algorithm [10 marks].

Algorithm 1 function Pesky(n)

```
Require: An integer $n \geq 0$.
$r \leftarrow 0$
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $i$ do
        for $k \leftarrow j$ to $i + j$ do
            $r \leftarrow r + 1$
        end for
    end for
end for
return $r$
```

   (a) What is the value returned by the function Pesky? Express your answer as a function of n and give the closed form.

   (b) Using $O()$ notation, give the worst-case running time of the function Pesky.

3. Consider the following recursive algorithm [10 marks].

Algorithm 2 function $g(n)$

```
Require: A positive integer $n$.
if $n \leq 1$ then
    return $n$
else
    return $5 \cdot g(n-1) - 6 \cdot g(n-2)$
end if
```

   (a) Set up a recurrence for this function's values and solve it to determine what this algorithm computes.

   (b) Set up and solve a recurrence relation for the number of multiplications made by this algorithm.

4. Let $T(n)$ be defined recursively as [5 marks]:

$$T(n) = \begin{cases} 4 & \text{if } n = 1 \\ T(n-1) + 4 & \text{otherwise} \end{cases}$$

   Show *by induction* that $T(n) = 4n$.

5. You're at the base of a staircase consisting of n stairs. If each step you make takes you either one or two stairs higher, what is the number of different ways in which you can climb the staircase? Explain how you derived your mathematical expression in words. [10 marks]

6. Design a divide-and-conquer algorithm for computing the number of levels in a binary tree. In particular, the algorithm should return 0 and 1 for the empty and single-node trees, respectively. What is the efficiency class of your algorithm? [10 marks]

7. Given a pristine chocolate bar with $n \times m$ pieces making up the bar. You need to break it into nm 1×1 pieces to share with nm people. You can break the bar only in a straight line, and once broken, only one piece at a time can be further broken. Design an algorithm that solves the problem with the minimum number of bar breaks. What is this minimum number? Justify your answer by using properties of a binary tree. (*Hint:* Think divide-and-conquer.) [10 marks]

8. Read the notes on OpenMP (file `OpenMP.pdf`) compiling and executing the sample programs included in the zip file (file `openmp.zip`) as you go to familiarize yourself with the various OpenMP directives. Take screenshots of each activity as you work your way through the exercise to include in your solution, including answers to any questions asked. [20 marks]

   Answer the following questions; it is expected that you will run each program multiple times. You may need to modify the program(s) to answer the questions. Be sure to explain the experiments you conducted.

   (a) For the program in the file `parfor.cc` vary the value of the variable `n` and the number of threads specified in the `num_threads` clause. How are the iterations distributed among threads? Be sure to try out fewer iterations than threads, and more iterations than threads, etc.

   (b) For the program in the file `private.cc` explore the values of the variables `a` and `i` before, after, and inside the parallel region. Try initializing the variables before the `#pragma` and also not initializing them.

   (c) For the program in the file `reduction.cc` explore the run time of the reduction clause by varying the number of threads in the `num_threads` clause. (Thinking ahead: Can you use a reduction to implement the parallel sum in the `ParallelAverage` command in Project #1?)

   (d) For the program in the file `schedule.cc` explore the impact of the schedule `kind` and `chunk_size` on how chunks are assigned to threads. Investigate the schedules when `kind` is `static`, `dynamic`, and `runtime`. Specifically, for `n=200` iterations and `num_threads(5)` plot a graph for each schedule `kind`: give the iteration number along the x-axis $(1, \ldots, 200)$ and the thread identifier along the y-axis $(0, \ldots, 4)$. This graph should indicate which thread was assigned to which loop iteration, graphically illustrating the differences between the kinds of schedules. Explain the results of your graphs. (*Hint:* To make `schedule.cc` run faster, put the threads to sleep for less time.)