CSCI 3104 Fall 2021
Instructors: Profs. Grochow and Waggoner

Midterm 2- Standard 18

Due Date ..... Nov / 20^{th} / 2021
Name ..... **Michael Ghattas**
Student ID **109200649**

Contents

1 Instructions

- The solutions **should be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You **may not collaborate with other students**. **Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Discord, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

Problem 1.

- My submission is in my own words and reflects my understanding of the material.
- I have not collaborated with any other person.
- I have not posted to external services including, but not limited to Chegg, Discord, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

I agree to the above, Michael Ghattas. □

3 Standard 18- Divide and Conquer Principles

Problem 2. Consider the following divide-and-conquer algorithm for change-making with standard US coin denominations (1,5,10,25). Here we are assuming that the solution is simply returned as a list of the coins used, e.g. $[5, 1, 1, 1, 5, 1, 1, 1]$ would be a valid output in making change for 11 cents. Here, $[5, 1, 1, 1, 5, 1, 1, 1]$ indicates that we used a nickel, followed by three pennies, another nickel, and another three pennies.

```
MakeChange(n):
  if n is one of 1,5,10,25:
    return [n] // use one of the exact right coin
  else if n < 5:
    return a list of ones, of length n // use pennies
  else:
    return MakeChange(floor(n/2)) appended with MakeChange(ceil(n/2))
```

Discuss in 1-2 sentences why this algorithm doesn't always return the smallest number of coins, and give an example where this algorithm makes change using more coins than are necessary.

Answer:

Let's consider the following example:

- $n = 6$
- Based on the input, MakeChange skips the "if" and "else if" parts of the code and moves to the "else" part
- $n = 6$ is now split into two parts, $floor(3)$ and $ceiling(3)$, each call MakeChange
- From the "else if" part $floor(3)$ and $ceiling(3)$ each return [1, 1, 1], thus our first output elements are $[1, 1, 1, 1, 1, 1]$

Here we can see that the output $[1, 1, 1, 1, 1, 1]$, which is 6 single pennies for a total of 6 coins. This is not the the smallest number of coins, since clearly the smallest number of coins would have been a nickel and a penny $[5, 1]$ for a total of 2 coins.

The problem with this algorithm is that it tries to break down the input value into the smaller parts if it does not recognize it, instead of trying to test if the input value is devisable by each of the recognizable values through a *mod* function, starting with the largest recognizable value and moving down (25, 10, 5, 1). Once an input value is found devisable by a recognizable value the recognized value is stored as part of the output, and then it deals with the reminder the same way through recursion till there is no reminder left, and the final output is completed accordingly.

□