Homework 5: Induction

Due date: Wednesday July 30th at 11:59 PM
If you work with others (and you should!), remember to follow the collaboration policy outlined in the syllabus. In general, you are graded on both the clarity and accuracy of your work. Your solution should be clear enough that someone in the class who had not seen the problem before would understand it.

We sometimes describe approximately how long our explanations are. These are intended to help you understand approximately how much detail we are expecting. You are allowed to have longer explanations, but explanations significantly longer than necessary may receive deductions.

In order to assist with writing English proofs, we've published a style guide on the website containing some tips. This guide contains references to proof materials that we haven't taught yet, so don't worry if some of these terms are unfamiliar.

Finally, be sure to read the grading guidelines for more information on what we're looking for.

1. Real World: Some Simple Code

Computer scientists write code, but if you want other people to use your code, you'll have to be able to explain why it works. In future classes (especially CSE 421), you'll do that with a proof.[1]

Let's take an example of code that returns the product of the first n even integers for inputs of $n > 0$.[2]

```
public static int multiplyEvens(int n) {
    if (n <= 0) {
        throw new IllegalArgumentException();
    }
    if (n == 1) {
        return 2;
    }
    return (2 * n) * multiplyEvens(n-1);
}
```

Call	Output	Reason
`multiplyEvens(1);`	2	2 = 2
`multiplyEvens(2);`	8	2 * 4 = 8
`multiplyEvens(3);`	48	2 * 4 * 6 = 48
`multiplyEvens(4);`	384	2 * 4 * 6 * 8 = 384

In this problem, you will write a correctness proof that will show that this code always returns the product of the first n even integers for inputs of $n > 0$. In other words that

$$\texttt{multiplyEvens(n)} = \prod_{i=1}^{n} 2i$$

Capital pi ($\prod$) is the product symbol which represents repeated multiplication, similar to $\sum$ which is for repeated summation. For example, $\prod_{k=1}^{3} 4k = 4 \cdot 8 \cdot 12$.

(a) Prove that the code produces the desired output. You must use induction for this problem. Be sure to start by defining your predicate $P()$.

(b) Take a moment to reflect on the structure of the proof and the code. The code has a base case and a recursive case, which relies on the result for input $n-1$ to calculate the answer for input n. The proof will have a base case, and an inductive case from k to $k+1$. You do not have to write anything for this part

[1] In the real world, you won't do a full proof (unless you become a researcher), but you will still have to clearly explain what's going on in your code, and a proof is a good way to practice a careful explanation

[2] We took this coding problem from `Practice-It`; It was authored by Whitaker Brand (on 2019/09/19)

2. Journey to the End

After fighting your way through the Overworld and the Nether, you've finally gathered all the supplies you need to fight the Ender Dragon. Depending on which End portal you go to, you need to figure out just how much food you should be bringing for the trek. You can bring two different foods: cooked cod or entire cakes. Cooked cod gives you 5 food points, cakes give you 14 (this is assuming you have to eat the entire cake at once). You're trying to be low waste and want to bring exactly the right amount of food for your journey. Prove that for every integer $n \geq 52$, there is some distribution of cooked cod and cakes totaling n food points such that you have exactly enough for your journey.

This question is based on Minecraft.

3. f(unky) Induction

Suppose we have the following recursively defined function,

$$f(n) = \begin{cases} 1 & \text{if } n = 0 \\ 3 & \text{if } n = 1 \\ (9n^2 - 9n) \cdot f(n-2) & \text{otherwise} \end{cases}$$

Use induction to prove that for all integers n with $n \geq 0$, $f(n) = 3^n n!$.
Recall that for a positive integer n, $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1$, and that $0! = 1$.

4. Running Times

You wrote a piece of recursive code. On an input of size n (for $n \in \mathbb{Z}^+$), your function takes $T(n)$ time to run, where:

$$\begin{aligned} T(n) &= 5n && \text{if } 1 \leq n \leq 4 \\ T(n) &= T(\lfloor n/2 \rfloor) + T(\lfloor n/4 \rfloor) + 5n && \text{for all } n > 4 \end{aligned}$$

In the definition above, $\lfloor x \rfloor$ is the "floor" function, it returns the greatest integer at most x.
For example: $\lfloor 3.2 \rfloor = 3$, $\lfloor 3.7 \rfloor = 3$, $\lfloor 3 \rfloor = 3$.

Show that for all $n \in \mathbb{N}$ with $n \geq 1$, $T(n) \leq 20n$

Caution: Your $T()$ is only defined to take positive integers as input. $T(1.5)$ is not defined!

Hint 1: Notice that while $T()$ is defined with equality, you are only proving an inequality.
Hint 2: The only fact about the floor function you will need is $\lfloor x \rfloor$ is an integer and $1 \leq \lfloor x \rfloor \leq x$.

5. Induction Rocks! [Extra Credit]

Consider an infinite sequence of positions $1, 2, 3, \ldots$ and suppose we have a stone at position 1 and another stone at position 2. In each step, we choose one of the stones and move it according to the following rule: Say we decide to move the stone at position i; if the other stone is not at any of the positions $i+1, i+2, \ldots, 2i$, then it goes to $2i$, otherwise it goes to $2i+1$.

For example, in the first step, if we move the stone at position 1, it will go to 3 and if we move the stone at position 2 it will go to 4. Note: no matter how we move the stones, they will never be at the same position.

Use induction to prove that, for any given positive integer n, it is possible to move one of the stones to position n. For example, if $n = 7$ first we move the stone at position 1 to 3. Then, we move the stone at position 2 to 5 Finally, we move the stone at position 3 to 7.

6. Feedback [Extra Credit]

Answer these questions on the separate gradescope box for this question.

Please keep track of how much time you spend on this homework and answer the following questions. This can help us calibrate future assignments and future iterations of the course, and can help you identify which areas are most challenging for you.

- How many hours did you spend working on this assignment (excluding any extra credit questions, if applicable)? Report your estimate to the nearest hour.
- Which problem did you spend the most time on?
- Any other feedback for us?