Final take-home assessment

CSC236, Winter 2020

Instructions:

- Unless otherwise instructed, you have until April 9 at 11:59pm to submit your solutions.
- Solutions should be submited to MarkUs.
- You don't need to use LaTeX, but we would prefer that you upload your solutions as a pdf. Please avoid proprietary formats such as docx.
- This final is open-book. You may consult your notes and official materials including anything linked from the course website. Do not try to search for solutions online.
- Absolutely no discussion or collaboration is allowed on this assessment.
- Clarifying questions can be sent to the instructor via email or private Piazza questions. You should limit yourself to the kinds of questions you would ask in a test setting. We won't be giving you hints or helping you with course material.
- If you're not sure how to solve a problem, you can still earn part marks for having the correct proof structure.

1. [10 marks]

   Suppose P is a predicate, and we know that whenever $P(n+1)$ is true, $P(n)$ is also true: in other words, suppose

   $$\forall n \in \mathbb{N}.\, (P(n+1) \implies P(n))$$

   Prove that for every natural number m, if $P(m)$ is true, then $P(k)$ is true for all natural numbers $k < m$. For example, if $P(2)$ is true, then $P(1)$ and $P(0)$ must also be true.

   Your answer should be a formal proof using some form of induction (either simple induction, complete induction, or the principle of well-ordering).

   Note: if you write an argument like the following, you will get 0 marks, because it is not written as an induction or well-ordering proof (even though the idea behind it is correct): "If $P(m)$ is true, $P(m-1)$ is also true (since $P(m) \implies P(m-1)$), and similarly $P(m-2)$ must be true since $P(m-1) \implies P(m-2)$, and so on. In this way, we can see that $P(m-1), P(m-2), \ldots, P(0)$ are all true."

2. [9 marks]

   Consider the following Python function:

```
1 def f(n):
2   if n < 3:
3     return n
4   else:
5     return f(n-3)
```

   Precondition: `n` $\in \mathbb{N}$

   Postcondition: Returns a natural number.

   For $n \in \mathbb{N}$, let $T(n)$ be the total number of times `f` gets called when `f(n)` is run (including the initial call). For example, $T(2) = 1$ since `f(2)` never makes any recursive calls, and $T(6) = 3$ since `f(6)` calls `f(3)` which calls `f(0)`.

   (a) Write a recurrence for $T(n)$. Briefly justify your recurrence in 1–2 sentences.

   (b) Write a closed form for $T(n)$.
   (You don't need to show how you found the closed form.)

   (c) Use complete induction to prove that your closed form in part (b) is the correct solution to your recurrence from part (a).

3. [20 marks]

   Given a list of 0's and 1's, A, we will say that A is **balanced** if it contains exactly as many zeroes as ones. For example, [ ] and $[0, 1, 0, 1]$ are balanced, but $[0, 1, 0]$ is not. Consider the Python function `bal`, defined below, which decides whether a list is balanced.

```
def bal(A):
    i = 0
    j = 1
    # Make a copy of A
    B = A.copy()
    while i < len(B) and j < len(B):
        if B[i] == B[j]:
            j += 1
        else:
            # swap the elements at indices j and i+1
            B[j], B[i+1] = B[i+1], B[j]
            i = i + 2
            j = i + 1
    return i == len(B)
```

Precondition: `A` is a list containing only 0's and 1's.

Postcondition: Returns True iff `A` is balanced.

(a) Use induction to prove that at the end of each iteration of bal's while loop, the sublist $B[:i]$ is balanced. (Recall that $B[:i]$ is a list containing all the elements of B up to but not including index i. If $i \geq \text{len}(B)$, then $B[:i] = B$)

You may need to strengthen your induction by proving additional invariants.

(b) Prove that `bal` is partially correct. State any loop invariants you want to use at the start of your answer. You may assume those invariants without proof.

You should be confident that these invariants are actually true - if your proof relies on an invalid invariant, you may lose significant marks. (You should also avoid contrived invariants that trivially entail partial correctness. For example, 'at the end of each iteration k, if $i_k \geq \text{len}(B)$ or $j_k \geq \text{len}(B)$, then A is balanced if and only if i is equal to $\text{len}(B)$'. Or, even more to the point, 'at the end of each iteration k, `bal` is partially correct'.)

4. [8 marks]

Let $\Sigma = \{\texttt{0}, \texttt{1}, \texttt{2}, \texttt{3}, \texttt{4}, \texttt{5}, \texttt{6}, \texttt{7}, \texttt{8}, \texttt{9}, \times, \texttt{=}\}$, so strings in Σ^* can have digits, as well as the symbols $\times$ and `=`. Let $\text{EQ} \subseteq \Sigma^*$ be the set of true equations of the form $a\times b\texttt{=}c$, where a, b and c are strings of digits representing base-10 numbers and $\times$ is multiplication. For example:

- $\texttt{2}\times\texttt{3=6} \in \text{EQ}$
- $\texttt{2}\times\texttt{03=00006} \in \text{EQ}$ (leading zeros are ignored)
- $\texttt{100}\times\texttt{37=3700} \in \text{EQ}$
- $\texttt{2}\times\texttt{3=7} \notin \text{EQ}$, since $2 \cdot 3 \neq 7$.

- $\texttt{1}\times\texttt{=1=1=}\times \notin$ EQ, since it doesn't have the form $a\times b\texttt{=}c$.

Prove that EQ is not regular.

5. [21 marks]

   Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and let S be the smallest subset of Σ^* satisfying:

   - Base case: $\varepsilon \in S$.
   - First constructor case: If $x \in S$ then $x\texttt{aa} \in S$.
   - Second constructor case: If $x, y \in S$ then $x\texttt{b}y \in S$.

   (a) List two strings of length 4 that are in S and two strings of length 4 that are in $\{\texttt{a}, \texttt{b}\}^*$ but not in S. (You don't need to explain your answer.)

   (b) Describe the set S in English. Don't just translate the definition — you should understand what strings are in S. In particular, your answer to this question should make it easy to understand why your answers to parts (e) and (f) are correct.

   (For example, if the base case were $\varepsilon \in S$ and the constructor cases were $\texttt{a}x \in S$ and $x\texttt{b} \in S$, a good description would be: "strings made of zero or more a's followed by zero or more b's".)

   (c) Write a regular expression denoting S. You don't need to explain your answer.

   (d) Prove that your description in part (b) is correct: that every string in Σ^* is in S if and only if it matches your description.

   (e) Draw a DFSA that accepts S.

   (f) For each state q in your DFSA from part (e), describe the set $L_q = \{x \in \{\texttt{a}, \texttt{b}\}^* \mid \delta^*(s, x) = q\}$ where s is your starting state.

   (In other words, for each state, describe set of inputs that cause your DFSA to finish in that state.)