Name: ______________________

Entry number: ______________________

There are 2 questions for a total of 10 points.

---

1. (5 points) Prove or disprove: Let $f : \mathbb{Z}^+ \to \mathbb{R}^+$ and $g : \mathbb{Z}^+ \to \mathbb{R}^+$ be any functions such that:

   1. $f(n)$ is $O(g(n))$, and
   2. p and q are functions mapping $\mathbb{Z}^+$ to $\mathbb{R}^+$ where $p(n) = \log f(n)$ and $q(n) = \log g(n)$.

   Then $p(n)$ is $O(q(n))$.

Solution: we will disprove the above statement using a counterexample. Consider the following two functions:

- $f(n) = 2$
- $g(n) = 1 + \frac{1}{2^n}$

First, we need to check that the above two functions satisfy conditions (1) and (2).

Claim 1: $f(n)$ is $O(g(n))$

Proof. Consider the constants $c = 2$ and $n_0 = 1$. For these constants we have that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. So, $f(n)$ is $O(g(n))$. □

Claim 2: p and q are functions mapping $\mathbb{Z}^+$ to $\mathbb{R}^+$ where $p(n) = \log f(n)$ and $q(n) = \log g(n)$.

Proof. $p(n) = \log f(n) = 1$ and $q(n) = \log g(n) = \log\left(1 + \frac{1}{2^n}\right)$. Both these functions map positive integers to positive real numbers. □

Finally, we show that $p(n)$ is not $O(q(n))$.

Claim 3: $p(n)$ is not $O(q(n))$.

Proof. We will show that for every constant c, n_0 there is a $n \geq n_0$ such that $p(n) > c \cdot g(n)$. Consider $n' = \log\left(\frac{1}{2^{1/c}-1}\right) + 1$. Note that for all $n \geq n'$ we have $1 > c \cdot \log\left(1 + 1/2^n\right)$ which means that for all $n \geq n', p(n) > c \cdot q(n)$. Now, consider $n'' = \max(n', n_0)$. Then clearly, $n'' \geq n_0$ and $p(n'') > c \cdot q(n'')$ which completes the proof of the claim. □

2. (5 points) Consider the following problem:

   SAME-BEHAVIOUR: Given descriptions $\langle A \rangle, \langle B \rangle$ of algorithms A and B respectively, determine if the behaviour of algorithms A and B are the same on all inputs.

   (*Algorithms A and B are said to have the same behaviour on input x, if either they both halt (exclusive-)or both do not halt.*)

   An algorithm P is said to solve the above problem if $P(\langle A \rangle, \langle B \rangle)$ halts and outputs 1 when A and B have the same behaviour on all inputs, and it halts and outputs 0 otherwise.

   Prove: There does not exist an algorithm P that solves the problem SAME-BEHAVIOUR.

Solution: In the class we have shown that there does not exist an algorithm for the halting problem.

HALTING: Given the description $\langle A \rangle$ of an algorithm A and an input x, determine if A halts on input x.

We will prove the statement by contradiction. We will argue that if there exists an algorithm for the above problem, then there also exists an algorithm for the halting problem. This implies that such an algorithm cannot exist. Indeed, suppose for the sake of contradiction there exists an algorithm P that determines if the behaviour of two given algorithms are the same. We will construct an algorithm Q for the halting problem that uses P as a subroutine. First, we need to define an algorithm $B_{A,x}$ with respect to another algorithm A and input string x.

$B_{A,x}(y)$
- if $(x == y)$ then halt
- else execute $A(y)$

Here is the description of the algorithm Q for the halting problem.

$Q(\langle A \rangle, x)$
- return$(P(\langle A \rangle, \langle B_{A,x} \rangle))$

It would be sufficient to show that Q outputs 1 iff A halts on x.

Claim 1: $Q(\langle A \rangle, x) = 1$ iff A halts on x.

Proof. Since there is an iff there are two directions to prove.

First we argue that if A halts on x, then Q outputs 1. This is true since from definition of $B_{A,x}$, the behaviour of A and $B_{A,x}$ is the same on all inputs except x. Moreover, on input x both A and $B_{A,x}$ halts. So, A and $B_{A,x}$ have the same behaviour on all inputs. In this case, P outputs 1 and this is what Q returns.

Next, we argue that if A does not halt on x, then Q does not output 1. This is true since the behaviour of A and $B_{A,x}$ differs on input x and hence P outputs 0 which is returned by Q. $\square$