PLEASE HAND IN

UNIVERSITY OF TORONTO
FACULTY OF ARTS AND SCIENCE

TERM TEST #2

CSC 236H1
DURATION — 50 MINUTES

PLEASE HAND IN

LAST NAME: ______________________________

FIRST NAME: ______________________________

Do NOT *turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below.)

This test consists of 2 questions on 4 pages (including this one). *When you receive the signal to start, please make sure that your copy of the test is complete.*

Please answer questions in the space provided. You will earn 20% for any question you leave blank or write "I cannot answer this question," on. You will earn substantial part marks for writing down the outline of a solution and indicating which steps are missing.

Good Luck!

QUESTION 1. [8 MARKS]

Read over this short recursive program and its pre- and post-condition. Then prove that if `pow(x, y)` is executed with any x and y that satisfy the precondition, the postcondition must be satisfied. You should prove this using complete induction on y. You may assume, without proof, that for any natural number y, $y = 2(y//2) + (y\%2)$, and that for any natural number y greater than 1, $0 \leq (y//2), (y\%2) < y$.

$y//2$ means y integer-divided by 2, or $\lfloor y/2 \rfloor$.

```
def pow(x, y) :
    if y == 0 : return 1
    elif y == 1 : return x
    else :
        return pow(x, y//2) * pow(x, y//2) * pow(x, y%2)
```

PRECONDITION: $x \in \mathbb{R}, y \in \mathbb{N}$

POSTCONDITION: `pow(x, y)` terminates and returns x^y.

SAMPLE SOLUTION: Define $P(y)$: as "If $x \in \mathbb{R}$ and `pow(x,y)` is executed, then it terminates and returns x^y." I prove that $\forall y \in \mathbb{N}, P(y)$ using complete induction.

INDUCTION STEP: Assume y is an arbitrary natural number, and that $P(i)$ is true for natural numbers $0 \leq i < y$. There are two cases to consider.

CASE 1, $y < 2$: In these cases, I can verify the claim directly. $P(0)$ claims that if x is a real number, then `pow(x,0)` returns $x^0 = 1$, and this is clearly what occurs, since `y == 0` satisfies the if statement on the first line. $P(1)$ claims that if x is a real number, then `pow(x, 1)` returns $x^1 = x$, and this is clearly what occurs, since $y \neq 0$ falsifies the if statement on the first line, and then satisfies the `elif` test on the second line. So, in both cases, $P(y)$ holds.

CASE 2, $y > 1$: Since $0 \leq (y//2), (y\%2) < y$, the induction hypothesis assumes $P(y//2)$ and $P(y\%2)$. Also, $y > 1$ means that both the `if` and `elif` tests fail, so the `else` branch executes, returning:

$$\begin{aligned} \text{pow}(x, y//2) \times \text{pow}(x, y//2) \times \text{pow}(x, y\%2) &= x^{y//2} \times x^{y//2} \times x^{y\%2} \quad \text{\# by IH} \\ &= x^{y//2+y//2+y\%2} \quad \text{\# adding exponents} \\ &= x^y \quad \text{\# by assumption} \end{aligned}$$

So, $P(y)$ holds in this case also.

In all cases, if $y \in \mathbb{N}$ and $P(i)$ is true for natural numbers $0 \leq i < y$, then $P(y)$ follows.

I conclude, by the principle of complete induction, $\forall y \in \mathbb{N}, P(y)$.

Combining $\forall y \in \mathbb{N}, P(y)$ with the fact that I assumed only that $x \in \mathbb{R}$, this establishes that whenever $y \in \mathbb{N}$ and $x \in \mathbb{R}$ (i.e. the precondition is satisfied), then `pow(x,y)` terminates and returns x^y (the postcondition).

QUESTION 2. [8 MARKS]

The definition of pow suggests the following recurrence for its time complexity, based on the value of y:

$$T(y) = \begin{cases} 1 & \text{if } y < 2 \\ 1 + 2T(\lfloor y/2 \rfloor) \end{cases}$$

Notice that the recursive call `pow(x, y%2)` takes constant time with respect to y, so it is represented in the recurrence by 1.

PART (A) [2 MARKS]

Use the Master Theorem (reprinted for your reading pleasure at the end of this test) to find the complexity class of T. Show your work.

SAMPLE SOLUTION: $T(y)$ has the form of a recurrence covered by the Master Theorem, where $a = 2$ (number of recursive calls), $b = 2$ (number of pieces the problem is divided into), and $d = 0$ (degree of polynomial bounding the amount of work to divide and later re-combine the problem). According to the Master Theorem, this means that:

$$a = 2 > 1 = 2^0 = b^d$$

... so T is in complexity class $\theta(y^{\log_2 2}) = \theta(y)$, or linear in y.

PART (B) [2 MARKS]

What would the complexity class of T be if it represented just 1 recursive call `pow(x, y//2)` rather than 2? Show your work.

SAMPLE SOLUTION: Again, $T(y)$ has the form of a recurrence covered by the Master Theorem, except now $a = 1$, $b = 2$, and $d = 0$, so

$$a = 1 = 2^0 = b^0$$

... and T is in complexity class $\theta(y^0 \lg y) = \theta(\lg y)$.

PART (C) [4 MARKS]

Modify the definition of pow so that it returns the same result as before, but with just one recursive call `pow(x, y//2)`

SAMPLE SOLUTION: The idea is to avoid evaluating `pow(x, y//2)` twice, so that the cost of the recursive calculation is incurred only once. There are several ways to do this, and I choose to save the value

```
def pow(x, y) :
    if y == 0 : return 1
    elif y == 1 : return x
    else :
        p = pow(x, y//2)
        return p * p * pow(x, y%2)
```

MASTER THEOREM

Suppose a recurrence expressing the complexity of a divide-and-conquer algorithm has the following form:

$$T(n) = \begin{cases} k & \text{if } n \leq B \\ a_1 T(\lceil n/b \rceil) + a_2 T(\lfloor n/b \rfloor) + f(n) & \text{if } n > B \end{cases}$$

where $B, k > 0$, $a_1, a_2 \geq 0$, and $a_1 + a_2 > 0$, and $f(n)$ is the cost of splitting and recombining.
If $f \in \theta(n^d)$, and $a = a_1 + a_2$, then

$$T \in \begin{cases} \theta(n^d) & \text{if } a < b^d \\ \theta(n^d \log n) & \text{if } a = b^d \\ \theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

1: ______/ 8

2: ______/ 8

TOTAL: ______/16