Cmput 325 Quiz (Part of Assignment Two)

INSTRUCTIONS:

- This is an open book/notes quiz, which is considered part of Assignment Two. The mark total is 4 assignment marks.
- There is no restriction how much time you may use. The rule regarding collaboration is the same as for programming assignments. In particular, no matter how you discussed with others, you have to write your own solutions and be able to explain what you submitted.
- Type your answers into a file, which should be named `<yourStudentID.quiz1>` and submitted via eClass using the submission link of this part of the assignment.
- For the lambda symbol λ, you can use the capital letter L. E.g., $(\lambda xy \mid xy)$ can be written as $(Lxy \mid xy)$.

1. [1 mark] Reduce the following λ-expression to its normal form. Show two sequences of reductions, one of which is based on normal order reduction and the other based on applicative order reduction. Show all the reduction steps.

$$(\lambda xy \mid xx(yy))\ (\lambda x \mid xy)\ (\lambda x \mid x)$$

2. [1.5 marks] Recall in lambda calculus, logic connectives NOT and OR can be defined as:

$$NOT = (\lambda x \mid xFT) \qquad OR = (\lambda xy \mid xTy)$$

where $T = (\lambda xy \mid x)$ and $F = (\lambda xy \mid y)$.

In logic, that "x implies y" is written $x \supset y$ (or in some textbooks, $x \to y$ or $x \Rightarrow y$). Denote this function by **IMP**

(a) Give a lambda expression that defines **IMP**, i.e., write what is missing at the right hand side of the vertical bar below.

IMP $= (\lambda xy \mid ......)$

Make sure that your answer is a normal form, i.e., it cannot contain expressions that are still reducible.

Hint: In logic, we know $x \supset y \equiv \neg x \vee y$.

(b) Using your definition, for each expression below, reduce it to a normal form. Here, the order of reduction is unimportant.

- **IMP** TF
- **IMP** FT

3. [1.5 marks] In this question, we consider the interpreter based on context and closure. Notationally, we will use $\{x_1 \rightarrow v_1, ..., x_m \rightarrow v_m\}$ for context and $[Fn, CT]$ for closure, where Fn is a lambda function and CT is a context. We assume that the initial context is CT0.

Consider the following λ-expression:

```
(((lambda (x y) (lambda (z) (if (> x y) (+ x z) (+ x y)))) 4 5) 10)
```

- What is the result of evaluating this expression?
- What is the **last context** created during this evaluation? Please provide some sketches of how you arrived at your answer.