CSC165H1: Problem Set 1

Due January 24, 2018 before 10pm

General instructions

Please read the following instructions carefully before starting the problem set. They contain important information about general problem set expectations, problem set submission instructions, and reminders of course policies.

- Your problem sets are graded on both correctness and clarity of communication. Solutions that are technically correct but poorly written will not receive full marks. Please read over your solutions carefully before submitting them.
- Each problem set may be completed in groups of up to three. If you are working in a group for this problem set, please consult `https://github.com/MarkUsProject/Markus/wiki/Student_Groups` for a brief explanation of how to create a group on MarkUs.

  Exception: Problem Set 0 must be completed individually.
- Solutions must be typeset electronically, and submitted as a PDF with the correct filename. **Handwritten submissions will receive a grade of ZERO.**

  The required filename for this problem set is **problem_set1.pdf**.
- Problem sets must be submitted online through MarkUs. If you haven't used MarkUs before, give yourself plenty of time to figure it out, and ask for help if you need it! If you are working with a partner, you must form a group on MarkUs, and make one submission per group. "I didn't know how to use MarkUs" is not a valid excuse for submitting late work.
- Your submitted file should not be larger than 9MB. This may happen if you are using a word processing software like Microsoft Word; if it does, you should look into PDF compression tools to make your PDF smaller, although please make sure that your PDF is still legible before submitting!
- Submissions must be made *before* the due date on MarkUs. You may use *grace tokens* to extend the deadline; please see the Problem Set page for details on using grace tokens.
- The work you submit must be that of your group; you may not refer to or copy from the work of other groups, or external sources like websites or textbooks. You may, however, refer to any text from the Course Notes (or posted lecture notes), except when explicitly asked not to.

Additional instructions

- Final expressions must have negation symbols ($\neg$) applied **only** to predicates or propositional variables, e.g. $\neg p$ or $\neg Prime(x)$. To express "a is not equal to b," you can write $a \neq b$.
- When rewriting logical formulas into equivalent forms (e.g., simplifying a negated formula or removing implication operators), you must **show all of the simplification steps involved**, not just the final result. We are looking for correct use of the various simplification rules here.
- You may not define your own predicates or sets for this problem set; please work with the definitions provided in the questions.

1. **[6 marks] Propositional formulas.** For each of the following propositional formulas, find the following two items:

   (i) The truth table for the formula. (You don't need to show your work for calculating the rows of the table.)

   (ii) A logically equivalent formula that only uses the $\neg$, $\wedge$, and $\vee$ operators; *no* $\Rightarrow$ or $\Leftrightarrow$. (You *should* show your work in arriving at your final result. Make sure you're reviewed the "extra instructions" for this problem set carefully.)

   (a) **[3 marks]** $(p \Rightarrow q) \Rightarrow \neg q$.

   (b) **[3 marks]** $(p \Rightarrow \neg r) \wedge (\neg p \Rightarrow q)$.

2. **[8 marks] Fixed Points.**

Let f be a function from $\mathbb{N}$ to $\mathbb{N}$. A *fixed point* of f is an element $x \in \mathbb{N}$ such that $f(x) = x$. A *least fixed point* of f is the smallest number $x \in \mathbb{N}$ such that $f(x) = x$. A *greatest fixed point* of f is the largest number $x \in \mathbb{N}$ such that $f(x) = x$.

(a) **[1 mark]** Express using the language of predicate logic the English statement:

"f has a fixed point."

You may use an expression like "$f(x) =$ [something]" in your solution.

(b) **[2 marks]** Express using the language of predicate logic the English statement:

"f has a least fixed point."

You may use the predefined function f as well as the predefined predicates $=$ and $<$. You may not use any other predefined predicates.

(c) **[2 marks]** Express using the language of predicate logic the English statement:

"f has a greatest fixed point."

You may use the predefined function f as well as the predefined predicates $=$ and $<$. You may not use any other predefined predicates.

(d) **[3 marks]** Consider the function f from $\mathbb{N}$ to $\mathbb{N}$ defined as $f(x) = x \mod 7$.[1]
Answer the following questions by filling in the blanks.

The fixed points of f are: _____

The least fixed point of f is: _____

The greatest fixed point of f is: _____

[1] Here we are using the modulus operator. Given a natural number a and a positive integer b, $a \mod b$ is the natural number less than b that is the remainder when a is divided by b. In Python, the expression `a % b` may be used to compute the value of $a \mod b$.

3. **[6 marks] Partial Orders.** A binary predicate R on a set D is called a *partial order* if the following three properties hold:

   (1) (reflexive) $\forall d \in D,\ R(d, d)$

   (2) (transitive) $\forall d, d', d'' \in D,\ (R(d, d') \wedge R(d', d'')) \Rightarrow R(d, d'')$

   (3) (anti-symmetric) $\forall d, d' \in D,\ (R(d, d') \wedge R(d', d)) \Rightarrow d = d'$

   A binary predicate R on a set D is called a *total order* if it is a partial order and in addition the following property holds: $\forall d, d' \in D,\ R(d, d') \vee R(d', d)$.

   For example, here is a binary predicate R on the set $\{a, b, c, d\}$ that is a total order:

   $R(a, b) = R(a, c) = R(a, d) = R(b, c) = R(b, d) = R(c, d) = R(a, a) = R(b, b) = R(c, c) = R(d, d) =$ True and all other values are False.

   (a) **[2 marks]** Give an example of a binary predicate R on the set $\mathbb{N}$ that is a partial order but that is *not* a total order.

   (b) **[2 marks]** Let R be a partial order predicate on a set D. R specifies an ordering between elements in D. Whenever $R(d, d')$ is True, we will say that d is less than or equal to d', or that d' is greater than or equal to d. The following formula in predicate logic expresses that there exists a greatest element in D; that is, an element in D that is greater than or equal to every other element in D:

   $$\exists d \in D,\ \forall d' \in D,\ R(d', d)$$

   An element in D is said to be *maximal* if no other element in D is larger than this element. The following formula in predicate logic expresses that there exists a maximal element in D:

   $$\exists d \in D,\ \forall d' \in D,\ d = d' \vee \neg R(d, d')$$

   Give an example of a partial order order R over $\{a, b, c, d\}$ such that every element is maximal.

   (c) **[2 marks]** Give an example of a partial order R over $\{a, b, c, d\}$ such that $a \in D$ is maximal but a is not a greatest element. Justify your answer briefly.

4. **[13 marks] One-to-one functions.** So far, most of our predicates have had sets of numbers as their domains. But this is not always the case: we can define properties of any kind of object we want to study, including functions themselves!

   Let S and T be sets. We say that a function $f : S \to T$ is **one-to-one** if no two distinct inputs are mapped to the same output by f. For example, if $S = T = \mathbb{Z}$, the function $f_1(x) = x + 1$ is one-to-one, since every input x gets mapped to a distinct output. However, the function $f_2(x) = x^2$ is not one-to-one, since $f_2(1) = f_2(-1) = 1$. Formally we express "$f : S \to T$ is one-to-one" as: $\forall x_1 \in S,\ \forall x_2 \in S,\ f(x_1) = f(x_2) \Rightarrow x_1 = x_2$.

   We say that $f : S \to T$ is **onto** if every element in T gets mapped to by at least one element in S. The above function $f(x) = x + 1$ is onto over $\mathbb{Z}$ but is not onto over $\mathbb{N}$. Formally we express "$f : S \to T$ is onto" as:

   $$\forall y \in T,\ \exists x \in S,\ f(x) = y$$

   Let $t \in T$. We say that f **outputs** t if there exists $s \in S$ such that $f(s) = t$.

   (a) **[1 mark]** How many functions are there from $\{1, 2, 3\}$ to $\{a, b, c, d\}$?

   (b) **[1 mark]** How many one-to-one functions are there from $\{1, 2, 3\}$ to $\{a, b, c, d\}$?

   (c) **[1 mark]** How many onto functions are there from $\{1, 2, 3, 4\}$ to $\{a, b, c\}$?

   (d) **[2 marks]** Now let R be a binary predicate with domain $\mathbb{N} \times \mathbb{N}$. We say that R *represents* a function if, for every $x \in \mathbb{N}$, there exists a *unique* $y \in \mathbb{N}$, such that $R(x, y)$ (is True). In this case, we write expressions like $y = f(x)$.

   Define a predicate $Function(R)$, where R is a binary predicate with domain $\mathbb{N} \times \mathbb{N}$, that expresses the English statement:

   "R represents a function."

   You may use the predicates $<, \leq, =, R$, but may not use any other predicate or function symbols.

   In parts (e)-(h) below, you may use the predicate $Function(R)$ (in addition to the predicates $<, \leq, =, R$) in your solution, but may not use any other predicate or function symbols.

   (e) **[2 marks]** Define a predicate that expresses the following English statement.

   "R represents an onto function."

   (f) **[2 marks]** Define a predicate that expresses the following English statement.

   "R represents a one-to-one function."

   (g) **[2 marks]** Define a predicate that expresses the following English statement.

   "R represents a function that outputs infinitely many elements of $\mathbb{N}$."

   (h) **[2 marks]** Now define a predicate that expresses the following English statement.

   "R represents a function that outputs all but finitely many elements of $\mathbb{N}$."