Due February 3, 2023 before 1pm

General instructions for Problem Sets

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.
- Solutions must be typeset 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**.
- Each problem set may be completed in groups of up to three—**except for Problem Set 0**. If you are working in a group for this problem set, please consult https://github.com/MarkUsProject/Markus/wiki/Student-Guide for a brief explanation of how to create a group on MarkUs.
- 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 one or more partner(s), you must form a group on MarkUs, and make one submission per group.
- Your submitted file(s) should not be larger than 19MB. You might exceed this limit if you use a word processor like Microsoft Word to create a PDF; in that case, you should look into PDF compression tools to make your PDF smaller, but 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 credits* to extend the deadline; please see the course syllabus for details on using grace credits.
- MarkUs may be slow when many students try to submit right before a deadline. **Aim to submit your work at least one hour before the deadline. It is your responsibility to meet the deadline.** You can submit your work more than once (and you are encouraged to do so); the most recent version submitted within the deadline (or within the late submission period) is the version that will be marked.
- The work you submit must be that of your group; you may not use or copy from the work of other groups, or external sources like websites or textbooks. Please see the section on Academic Integrity in the course syllabus for further details.

Additional instructions

- You may not define your own propositional operators or predicates for this problem set unless asked for in the question. Please work with the ones we have introduced in lecture, and any additional definitions provided in the questions themselves.
- Final expressions in predicate logic 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 $a \neq b$.

- Please review the section *Our conventions for writing formulas* on pp. 31–33 of the Course Notes, and in particular the precedence rules for the different operators and quantifiers.
- 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.

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 do not need to simplify your formula. (You *should* show your work in arriving at your final result. Make sure you're reviewed the "Additional instructions" for this problem set carefully.)

   (a) $(\neg p \Leftrightarrow q) \Rightarrow q$.

   (b) $\big(p \Rightarrow (q \Rightarrow r)\big) \Rightarrow \big((p \Rightarrow q) \Rightarrow r\big)$.

2. **[12 marks] Translating statements.**

Consider the following sets and predicates:

Symbol	**Definition**
S	the set of all students
C	the set of all courses
$Study(s, c)$	"student s is taking course c this term," where $s \in S$ and $c \in C$
$Fail(s, c)$	"student s has failed course c," where $s \in S$ and $c \in C$
$CS(s)$	"student s is a Computer Science student" where $s \in S$

Using **only** these sets and predicates, the symbols $=$ and $\neq$, and the standard propositional operators and quantifiers from lecture, translate each of the following English statements into predicate logic.

(a) No student in Computer Science has failed any course.

(b) There is a course that all Computer Science students are taking this term.

(c) Not all students in Computer Science are taking courses this term.

(d) Every course has exactly 1 student. (Hint: use $=$ and/or $\neq$.)

(e) Every student is taking exactly two different courses this term. (Hint: use $=$ and/or $\neq$.)

Hint: Be sure to read the "Additional instructions" given on pages 1 and 2.

3. **[9 marks] Choosing a universe and predicates.** This question gets you to investigate some of the subtleties of variable scope and precedence rules that are discussed in pp. 31–33 in the Course Notes.

   (a) Consider the following two statements:

$$\forall x \in \mathbb{N},\ P(x, 165) \vee P(x, 0) \qquad \text{(Statement 1)}$$

$$\big(\forall x \in \mathbb{N},\ P(x, 165)\big) \Rightarrow \big(\exists x \in \mathbb{N},\ P(x, 0)\big) \qquad \text{(Statement 2)}$$

   Provide a definition of a binary predicate P, where each parameter has domain $\mathbb{N}$, that makes one of the above statements True and the other statement False. Your predicate may *not* be a constant function (i.e., always True or always False).

   Briefly justify your response, but no formal proofs are necessary.

   (b) Consider once again Statements 1 and 2 from Part (a).

   Provide a definition of a binary predicate P, where each parameter has domain $\mathbb{N}$, that makes **both** of the statements True. Your predicate may *not* be a constant function (i.e., always True or always False).

   Briefly justify your response, but no formal proofs are necessary.

   (c) Consider the following two statements:

$$\forall x \in S,\ \Big(\exists y \in T,\ P(x, y)\Big) \Rightarrow Q(x) \qquad \text{(Statement 3)}$$

$$\forall x \in S,\ Q(x) \Rightarrow \Big(\forall y \in T,\ P(x, y)\Big) \qquad \text{(Statement 4)}$$

   Provide a definition of *non-empty* sets S and T, a binary predicate P and a unary predicate Q, that makes one of the above statements True and the other statement False. The first parameter of P has domain S and the second parameter of P has domain T. The parameter of Q has domain S. Your sets must be non-empty, and your predicates may *not* be constant functions (i.e., always True or always False).

   Briefly justify your response, but no formal proofs are necessary.

4. **[8 marks] Pierre Numbers** A natural number n is said to be a "Pierre number" when it can be expressed as $2^{2^k} + 1$ for some integer k.

   (a) Write a predicate $PierreNumber(n)$ that is True if and only if n is a Pierre number.

   (b) Express the statement "All Pierre numbers are odd." using predicate logic. You may use the predicate $PierreNumber$, the predicate $=$ and logical operators, but may not use other predicates (like Odd or $Even$).

   (c) Express the statement "There is not a largest Pierre number." using predicate logic. You may use the predicate $PierreNumber$ and the standard inequality predicates, but may not use other predicates.

   (d) Express the statement "Only the five smallest Pierre numbers are prime." using predicate logic. You may use the predicate $Prime(n)$. The smallest Pierre number is 3. (Hint: you may find this question easier if you don't use your predicate $PierreNumber(n)$. Use the Pierre number definition instead.)