Exam 2

Read this page and fill in your name, pledge, and email ID now.

Name: ______________________________

Email ID: ______________________________

For this exam, you must **work alone**. You are not permitted to obtain help from people other than asking clarifying questions of the course staff. You are not permitted to provide help to others taking the exam. **You may not use any resources other than the one page of notes you prepared.**

Sign below to indicate that you understand these expectations and can be trusted to behave honorably:

Signed: ______________________________

The exam has 8 questions. For each question, there is ample space provided to hold an excellent answer. If you need more space, you can use the backs of pages, but include clear markings and arrows to indicate the answer that should be graded. We will assume anything not inside the answer space or clearly marked from it, is your scratch work that should not be considered in scoring your answers.

Do not open past this page until instructed to do so.

Revisiting Exam 1

1. Prove by induction that every finite non-empty subset of the real numbers contains a *least* element, where an element $x \in S$ is defined as the least element if $\forall z \in S - \{x\}. \, x < z$. (Note: you should not just assume all finite sets are well ordered for this question.)

State Machines and Invariants

2. For each of the statements below, indicate if the implication stated is valid (must always be true) or invalid (may be false). Provide a short justification supporting your answer.

a. If $M = (S, G : S \times S, q_0 \in S)$ is a state machine where G is a total surjective function, all states in S are reachable.

Circle one: *Valid* or *Invalid*

Justification:

b. If $P(q)$ is a *preserved invariant* for machine $M = (S, G, q_0)$, and r is a reachable state in M, $P(r)$ is true.

Circle one: *Valid* or *Invalid*

Justification:

c. If $M = (S, G, q_0)$ is a state machine and R is the set of reachable states in M,

$$S = R \implies (G \cup (q_0 \to q_0)) \text{ is an injective function.}$$

Circle one: *Valid* or *Invalid*

Justification:

Stable Matching

The state machine model of the Gale-Shapley algorithm from Class 15 and Problem Set 7 is shown (without modification) below.

$$
\begin{aligned}
S = \{ & (\textit{pairings}, \textit{proposals}) \mid \\
& \quad \textit{pairings} = \{(a,b) \mid a \in A, b \in B, \text{no duplicate occurences of } a \text{ or } b \text{ in } \textit{pairings}\}, \\
& \quad \textit{proposals} = (r_1, r_2, \cdots, r_n), 0 \le r_i \le n\} \\
G = \{ & (\textit{pairings}, \textit{proposals} = (r_1, r_2, \cdots, r_n)) \to (\textit{pairings}', \textit{proposals}') \mid \\
& \quad i \in \{1, \cdots, n\}, r_i < n \\
& \quad \forall b \in B.\, (a_i, b) \notin \textit{pairings} \\
& \quad b_x ::= \text{the } r_i\text{-ranked choice for } a_i \\
& \quad \textit{proposals}' = (r_1, \cdots, r_i + 1, \cdots, r_n) \\
& \quad \textit{pairings}' = \begin{cases} \textit{pairings} \cup \{(a_i, b_x)\} & \forall a_z \in A.\, (a_z, b_x) \notin \textit{pairings} \text{ (Case 1)} \\ \textit{pairings} \cup \{(a_i, b_x)\} - \{(a_z, b_x)\} & \exists a_z \in A.\, (a_z, b_x) \in \textit{pairings} \land a_z \prec_{b_x} a_i \text{ (Case 2)} \\ \textit{pairings} & \text{otherwise (Case 3)} \end{cases} \\
& \quad \} \\
q_0 = {} & (\textit{pairings} = \{\}, \textit{proposals} = (0, 0, \ldots, 0))
\end{aligned}
$$

3. For each of the following predicates, answer whether or not it is a *preserved invariant* for $M = (S, G, q_0)$ as defined above, and provide a brief justification supporting your answer. We use a, a_1, a_2 to denote elements of A, and b, b_1, b_2 to denote elements of B.

a. $P(q = (\textit{pairings}, \textit{proposals})) :=$ *pairings* contains a pair including a.

Circle one: *Preserved Invariant* or *Not Preserved*

Justification:

b. $P(q = (\textit{pairings}, \textit{proposals})) :=$ *pairings* contains a pair including b.

Circle one: *Preserved Invariant* or *Not Preserved*

Justification:

c. $P(q = (\textit{pairings}, \textit{proposals} = (r_1, r_2, \cdots, r_n))) := (r_1 + r_2 + \cdots + r_n) > 7.$

Circle one: *Preserved Invariant* or *Not Preserved*

Justification:

d. $P(q = (\textit{pairings}, \textit{proposals} = (r_1, r_2, \cdots, r_n))) := |\textit{pairings}| \leq (r_1 + r_2 + \cdots + r_n).$

Circle one: *Preserved Invariant* or *Not Preserved*

Justification:

For reference, here is the state machine repeated without change from previous page:

$$
\begin{aligned}
S = \{ & (\textit{pairings}, \textit{proposals}) \mid \\
& \textit{pairings} = \{(a, b) \mid a \in A, b \in B, \text{no duplicate occurences of } a \text{ or } b \text{ in } \textit{pairings}\}, \\
& \textit{proposals} = (r_1, r_2, \cdots, r_n), 0 \leq r_i \leq n\} \\
G = \{ & (\textit{pairings}, \textit{proposals} = (r_1, r_2, \cdots, r_n)) \to (\textit{pairings}', \textit{proposals}') \mid \\
& i \in \{1, \cdots, n\}, r_i < n \\
& \forall b \in B.\, (a_i, b) \notin \textit{pairings} \\
& b_x ::= \text{the } r_i\text{-ranked choice for } a_i \\
& \textit{proposals}' = (r_1, \cdots, r_i + 1, \cdots, r_n) \\
& \textit{pairings}' = \begin{cases} \textit{pairings} \cup \{(a_i, b_x)\} & \forall a_z \in A.\, (a_z, b_x) \notin \textit{pairings} \text{ (Case 1)} \\ \textit{pairings} \cup \{(a_i, b_x)\} - \{(a_z, b_x)\} & \exists a_z \in A.\, (a_z, b_x) \in \textit{pairings} \wedge a_z \prec_{b_x} a_i \text{ (Case 2)} \\ \textit{pairings} & \text{otherwise (Case 3)} \end{cases} \\
& \} \\
q_0 = & (\textit{pairings} = \{\}, \textit{proposals} = (0, 0, \ldots, 0))
\end{aligned}
$$

Recursive Data Types

Consider the recursive data type, *cist*, defined by:

– **Base: null** is a *cist*.
– **Constructor:** for all *cist* object, p, expand(p) is a *cist*.

The *size* of a *cist*, p, is the number of times it takes to expand starting from **null** to produce that *cist*. So, for example, size(expand(expand(expand(**null**)))) = 3.

4. Provide a precise and complete definition of *size* for *cist* all objects.

Define the merge of two cists as:

$$\text{merge}(p_1, p_2) = \begin{cases} p_2 & p_1 = \textbf{null} \\ \text{expand}(\text{merge}(q, p_2)) & p_1 = \text{expand(q)} \end{cases}$$

5. Prove that for any two cists, p_1 and p_2,

$$\text{size}(\text{merge}(p_1, p_2)) = \text{size}(p_1) + \text{size}(p_2).$$

Consider the Python program below, that returns the greatest element of the input list `p`. You may assume `p` is a non-empty list of natural numbers.

```
def max_element(p):
    x = p[0]
    i = 1
    while i < len(p):
        if p[i] > x:
            x = p[i]
        i = i + 1
    return x
```

6. Complete the definition of the state machine below that models `max_element`.

$$
\begin{aligned}
S &= \{(x,i) \mid x, i \in \mathbb{N}\} \\
G &= \{(x,i) \to (x',i') \mid \\
&\qquad i < \texttt{len(p)} \land \\
&\qquad i' = \underline{\hspace{5cm}} \ \land \\
&\qquad x' = \underline{\hspace{1cm}} \ \text{ if p[i]} > x \\
&\qquad x' = \underline{\hspace{1cm}} \ \text{ otherwise} \\
q_0 &= \underline{\hspace{5cm}}
\end{aligned}
$$

7. Prove that the state machine (from problem 6) always terminates.

8. Prove that `max_element`, as modeled by the state machine from Problem 6, always returns the maximum value of that list.

End of Exam Questions (last page is optional feedback).

Optional Feedback

This question is optional and will not negatively effect your grade.

Do you feel your performance on this exam will fairly reflect your understanding of the course material so far? If not, explain why. (Feel free to provide any other comments you want on the exam, or the course more broadly, here.)

Are there any things you hoped to learn in cs2102 that we haven't done yet?

Score: ____