The exam has 7 questions, equally weighted. You should not use any code or notes to help you with this exam. Please sign the Honor Pledge on the last page when you are finished with the exam. If you are not taking the exam with the rest of the class, please include your starting and stopping times when you sign the Honor Pledge.

1. **What does \( \text{lambda (x y) (+ x y)} \) parse to? What does it evaluate to in environment \( \text{env} \)?** In other words, if this expression parses to \( \text{tree} \) then what is \( \text{tree} \) and what is \( \text{eval-exp tree env} \)? Your answers to these need some datatypes. You can either give code for the datatypes or explain in English what fields they contain.
2. How could we extend MiniScheme to allow for unrestricted lambdas? For example, we might want to write

\[
(\text{let } ([\text{sum } (\lambda \text{nums} (\text{apply + nums}))])
\]

\[
(\text{sum 1 2 3})
\]

This expression should evaluate to 6. **How would you parse an unrestricted lambda?**

**How would you evaluate it in environment env?** Your answer can be in English or in code, but you should give enough detail that I could implement it.
3. We implemented if as a kind of MiniScheme expression:
   (if <condition> <true-branch> <false-branch>).
   We could also have implemented if as a primitive procedure with 3 arguments. How would primitive procedure if act differently than an if-expression?
4. a) Give an example of an expression that has a different value under dynamic binding than under static binding. Say what the expression evaluates to under each binding scheme.

b) Give an example of an expression that has a different value under call-by-reference than under call-by-value. Say what the expression evaluates to under each scheme.
5. We went to a lot of trouble to implement letrec. Why did we need to do that? **Explain why the following expression gives an error in standard Scheme:**

\[
\text{(let ([f (lambda (x) (if (= x 0) 1 (* x (f (- x 1))))])]
\text{(f 5)})}
\]
6. a) **What do people mean when they say an expression has “state”?** Your answer to this can be as short as one sentence.

b) **What would be different in the interpreter project if MiniScheme was stateless?**
7. I would like to add cond expressions to MiniScheme. Here is a typical cond expression:

```scheme
(cond
  [(< a b) (- b a)]
  [(< b a) (- a b)]
  [else 0])
```

So the word ‘cond is followed by a sequence of cases. Each case is a pair. If the first element of the pair is the word ‘else we return the value of the second element of the pair. If the first element of the pair evaluates to ‘True we return the value of the second element of the pair. Otherwise we go on to the next case. If there are no more cases just return `null. It is easy to parse such an expression into (‘cond-exp cases) where `cases is a list of pairs of parse trees. You can assume we have a datatype cond-exp with recognizer cond-exp? and getter cond-cases, and that the parsing is already completed.

**What code would you add to eval-exp to evaluate a cond-exp in environment env?**
Please write and sign the Honor Pledge when you are finished with the exam.