COMP 333 Final Practice Exam

The final exam is cumulative. This practice exam, **in addition to** the prior practice exams, assignments, in-class handouts, and exams, is intended to be a comprehensive guide for studying. This practice exam only focuses on material since the last exam. You are permitted to bring three 8.5 x 11 sheets of paper into the exam with you, as long as they have handwritten notes on them. Both sides of both sheets can be used. To be clear, these must be entirely handwritten.

Language Terminology / General Concepts

1.) Java and JavaScript both use garbage collection for memory management. In contrast, Rust uses ownership and borrowing.

1.a.) In 1-3 sentences, in your own words, explain how garbage collection reclaims memory. Your description doesn't have to be detailed enough to implement a garbage collector, only detailed enough to get the gist of when memory would be reclaimed.

1.b.) In 1-3 sentences, in your own words, explain how Rust's ownership and borrowing system reclaims memory. It doesn't need to be detailed enough to implement it, only detailed enough to know when memory would be reclaimed.

1.c.) Name one advantage of garbage collection over Rust's ownership/borrowing system.

1.d.) Name one advantage of Rust’s ownership/borrowing system over garbage collection.

2.) In 1-3 sentences, explain the difference between compilation and interpretation. Your answer does not need to be detailed enough to implement a compiler or interpreter.

3.) The Java Virtual Machine (JVM) is implemented as an interpreter over Java bytecode. Similarly, most JavaScript implementations are implemented as interpreters. However, most Java and JavaScript implementations support just-in-time (JIT) compilation.

3.a.) In 1-3 sentences, explain what JIT compilation does, in the context of an interpreter. Your answer doesn't need to be detailed enough to implement a JIT compiler.

3.b.) JIT compilers can sometimes generate faster code than traditional compilers. Why?

4.) C only has support for first-order functions, whereas JavaScript has support for higher-order functions.

4.a.) In 1-3 sentences, explain what higher-order functions are. You don't have to provide enough detail to explain how to use them.

4.b.) Unlike first-order functions, higher-order functions may require memory to be dynamically allocated at runtime. Why?

4.c.) Write a JavaScript code snippet that uses higher-order functions and would require memory to be dynamically allocated at runtime.

5.) Consider the following code snippet, which is written in some unknown programming language:

```
DefineFunction foo(x, y):
  DefineVariable temp = x dividedBy y
  return temp

foo(3, 4)             // first call to foo
foo("alpha", "beta") // second call to foo
```

5.a.) Assume this language is statically-typed. Does this code probably compile? Why or why not?

5.b.) Assume this language is dynamically-typed. Does this code probably compile? Why or why not?

6.) Consider the following Java function:

```
public static void foo() {
  int x = 0;
  Object obj1 = new Object();
  Object obj2 = obj1;
}
```

6.a.) If `foo` is called, what is allocated on the stack?

6.b.) If `foo` is called, what is allocated on the heap?

6.c.) When `foo` returns, what is guaranteed to be deallocated immediately?

6.d.) When `foo` returns, is there anything which might be deallocated at a later point? That is, what is *not* guaranteed to be immediately deallocated?

Rust

7.) Declare a struct named Example that holds five fields:

- `first`, which holds a `String`
- `second`, which holds a 32-bit unsigned integer
- `third`, which holds a 32-bit signed integer
- `fourth`, which holds a 64-bit unsigned integer
- `fifth`, which holds a 64-bit signed integer

8.) Consider the following Rust code:

```
fn main() {
    let s1: String = "foo".to_string();
    let s2: String = s1;
    println!("{}", s1);
}
```

This code does not compile, and the compiler produces an error message pointing to the use of `s1` in the `println!` statement. Why doesn't this code compile?