Banner number: Name:

Final Exam

CSCI 3136: Principles of Programming Languages

April 19, 2013

Group 1		Group 2		Group 3		$\sum$
Question 1.1		Question 2.1		Question 3.1		
Question 1.2		Question 2.2		Question 3.2		
Question 1.3		Question 2.3				
Question 1.4		Question 2.4				
$\sum$		$\sum$		$\sum$		

Information and instructions:

- The questions are divided into three groups. The questions in the first group ask you to explain some basic concepts in programming language design. The questions in the second group ask you to demonstrate your understanding of the basic principles underlying certain programming language features. The questions in the third group are problem solving questions. Make sure you allocate sufficient time to the questions in this latter group.
- Provide your answer in the box after each question. If you absolutely need extra space, use the backs of the pages, but try to avoid it. The size of each box is an indication of the length of the answer I expect.
- **You are not allowed to use a cheat sheet.**
- **Read every question carefully before answering.**
- **Do not forget to write your banner number and name on the top of this page.**
- **This exam has 12 pages, including this title page. Notify me immediately if your copy has fewer than 12 pages.**
- The total number of marks in this exam is 100.

1 Basic Concepts

Question 1.1 (Variables) — 10 marks

(a) Define the terms "L-value" and "R-value".

A variable can be used as an R-value or as an L-value. Explain the difference between the following two uses of variable x in the following two assignments. What does "x" refer to in each assignments?

```
x = 1;                    y = 2 * x;
```

(b) What does it mean for a language to use a "value model" or "reference model" for its variables? Define the two terms.

Does C use a value model or reference model?

Does Java use a value model or reference model?

Question 1.2 (Function parameters) — 10 marks

(a) Explain the following parameter passing modes.

By value

By reference

By sharing

(b) What are "optional function parameters" and "named function parameters"?

Is there a run-time cost to using optional or named function parameters as opposed to using "normal" positional parameters? Explain.

Question 1.3 (Object lifetime and memory management) **10 marks**

(a) We identified three different regions of a program's address space where objects can be stored. Where an object is stored depends mostly on its lifetime. Name the three different regions and for each explain what the lifetime of an object needs to be for the object to be stored in this region.

(b) Explain under what circumstances a static object may be stored on the heap.

(c) Explain the terms "internal fragmentation" and "external fragmentation".

Question 1.4 (Names, binding, and scope) 10 marks

(a) What is a “name”?

(b) What is a “binding”?

(c) Explain the terms “static binding” and “dynamic binding”.

(d) What is a “(stack) frame”?

(e) What is a “referencing environment”?

(f) What is a “closure”?

2 How Is It Done?

Question 2.1 (Heap management) 7 marks

Since objects stored on the heap can be allocated and deallocated at arbitrary times in a program, the run-time environment needs to track the free heap space. When the program makes a request for a heap block of a certain size, it has to be decided which part of the free heap space to allocate to the program. When a program releases an allocated heap block, the heap block needs to be added back to the free heap space. Describe an efficient method for managing the free heap space. “Efficient” means that the time per heap allocation/deallocation should be small and that the method can service allocation requests of arbitrary sizes while keeping internal fragmentation low.

Question 2.2 (Garbage collection or the next best thing) **13 marks**

(a) What problem do tomb stones and locks and keys address?

(b) Explain how tomb stones work.

(c) How do reference counts work?

(d) Can reference counts be considered a general garbage collection method? If so, argue that a garbage collector using reference counts always succeeds in deallocating objects that are no longer used. If not, give an example of a type of heap-allocated structure that a garbage collector using reference counts fails to deallocate even though it is no longer used.

Question 2.3 (Inheritance and dynamic method binding) **8 marks**

(a) Explain the difference between static and dynamic method binding in object-oriented languages.

(b) Explain how to implement single inheritance and dynamic method binding.

Question 2.4 (Iteration and recursion) **7 marks**

We discussed that every iterative procedure can be turned into a recursive procedure but that doing the reverse, while possible, is not easy. Consider the following iterative procedure. (I intentionally kept this question language-agnostic and provided pseudo-code. You should do the same in your answers.)

procedure SUM(A) **begin**
  $s := 0$
  for $i := 1$ **to** $|A|$ **do**
    $s := s + A[i]$
  end
  return s
end

(a) Translate this procedure into a recursive procedure.

(b) Is your procedure tail-recursive?

(c) If not, provide a tail-recursive version.

(d) What is the advantage of tail-recursive procedures over ones that are not tail-recursive in many languages?

3 Problems, Problems, Problems

Question 3.1 (Push-down automata) **10 marks**

Provide a graphical representation of a push-down automaton that recognizes the language

$$\{1^n0(01|10|11)^n \mid n \geq 0\}.$$

Do not forget to specify which mode of acceptance your PDA uses and which is the start symbol initially on the stack.

Question 3.2 (Parsing) 15 marks

Provide a context-free grammar for the language in Question 3.1, annotate each production with its PREDICT set, and provide a recursive-descent parser for the language based on these PREDICT sets.