**Review for COMP 141 Exam 1**

Section 1: History and Language Design

1. Describe two features of programming languages that the language Algol pioneered.
   1. Logos
   2. Procedures
   3. Introduced blocks
2. Describe the difference between compiled and interpreted languages. What are the pro’s and con’s of each?
3. Give 4 examples of abstractions that programming languages commonly provide and classify them as data abstractions or control abstractions.
4. Suppose you are designing a new language for beginning programmers to use in web development. Which 3 of the language design principles would be most important for your language and why?
5. Describe an example of a language (feature) that demonstrates either a good or bad application the following principles:
   1. Readability: C direct control over memory management,
   2. Writability: Java, high extensibility
   3. Portability: Java, virtual machine enables it to execute on multiple systems
   4. Security:
   5. Regularity: C++, most functions use {} to indicate end and begin
   6. Execution efficiency: Scheme highly abstract thus very good at execution efficiency
   7. Programmer efficiency: pseudo code enables Python good for programmer efficiency
   8. Reliability:
   9. Extensibility: Java
6. Describe the three stages of the translation process and give an example of an error that could be caught at each stage.
   1. Assembly: wrong key press
   2. Link: math errors
   3. Compile: logic errors
7. What is the von Neumann architecture and what features of imperative programming languages are designed to align with the von Neumann architecture?

The von Neumann architecture states that data and program stored in the same memory, the program follows a Fetch-Execute-Store cycle, and a single CPU executes instructions and modifies data

Imperative languages align with the von Neumann architecture by creating a memory area where programs and data are stored and a separate CPU that sequentially executes instructions fetched from memory

Section 2: Syntax

1. Define syntax.

Syntax is the set of rules, or grammar of a language, which make up what the computer interprets by the user and what is allowed.

1. The tokens of a language generally fall into four basic types: reserved words, literals, symbols and identifiers. Give two examples of each in C++, Java or Python.

Examples of reserved words in C++ are ‘bool’ and ‘while’

Examples of literals in C++ are things like ‘decimal numbers’ and ‘strings’

Examples of symbols in C++ are ‘+’ and ‘-‘

Examples of identifiers in C++ are ‘User defined variables’ and ‘Objects’

1. What is the principle of the longest substring? Use examples in C++, Python or Java to illustrate your answer.
2. How do free format languages differ from fixed format languages?
3. Give a regular expression for each of the following tokens in C++/Java:
   1. Variable
   2. Number (both integer and float)
4. Given the following terminal definitions, write an EBNF for a single dimensional array declaration in C++ (assume only primitive types).

IDENTIFIER: (\_|[a-z]|[A-Z])(\_|[a-z]|[A-Z]|[0-9])\*

INTEGER: ([0-9])+

1. Create a BNF grammar to represent the syntax of the condition portion of an if-statement.
   1. Is it possible to create an ambiguous statement in your grammar? If so, give an example.

<cond>:<bool-expr>|<bool-expr><comb><cpmd>

< bool-expr >: <val><comp><val>

<comp>:<|>|<=|>=|==|!=

<comb>: &&| ||

<val>:<var\_name>|<number>|<char>|<string>

An ambiguous statement: x < 5 and ( y < 5 or z < 5)

1. Consider the following EBNF for a loop.

<loop> 🡪 do <stmt> while \(<cond>{<log-op><cond>}\) ;

<cond> 🡪 <op><comp-op><op>

<op> 🡪 Number | Identifier

<stmt> 🡪 Identifier is (Identifier | Number); | <stmt>[<stmt>]

<log-op> 🡪 OR | AND

<comp-op> 🡪 < | > | == | !=

* 1. Give an example of a legal loop.
  2. Draw a parse tree of your answer.
  3. Convert the EBNF to BNF. You only need to re-write productions that are not in BNF form.

1. Write the C++ code to scan a line for tokens and classify them accordingly:

Var: a([a-z]|[A-Z]|!|@|#)\*

Num: [0-9]+

Op: + | - | \* | /

Eq: =

Section 3: Functional Programming and Scheme

1. What metaphor/abstraction is used for computation in the functional paradigm? Give 3 distinguishing features of the functional paradigm as compared to the imperative paradigm.
2. There are three different commands in Scheme that can be used to construct a list: list, cons and append. For each of the three commands, give an example of how that command could be used to create the list   
   (1 2 3).

(cons ‘1 ‘(2 3)) gives ‘(1 2 3)

(list ‘1 ’2 ‘3) gives ‘(1 2 3)

(append ‘(1) ‘(2) ‘(3)) gives ‘(1 2 3)

1. Write a function in Scheme called sumL that takes a list of numbers and returns the sum of the elements in the list.  
   (sumL ‘(1 2 3 4 5 6))  
   would return  
   21

(define (sumL L)

(if (null? L)

0

(+ (car L)(sumL (cdr L)))))

1. Write a Scheme function called count that takes an element and a list and returns the number of occurrences of element in the list. For example:  
   (count ‘a ‘(a b a c t a))  
   would return  
   3
2. Write a Scheme function called insert that takes an element and a list and returns the list with a copy of the element inserted between each item in the list. For example:  
   (insert ‘a ‘(bcd))  
   would return  
   (a b a c a d a)
3. Write a scheme function called sumX that takes an integer and finds the sum of all integers from that value down to zero. Make no assumptions about whether the number is positive or negative. For example:  
   (sumX -5)  
   would return  
   -10
4. Write a Scheme function called pay that takes a list containing an employee’s first name, wage and hours. The function should return the amount the employee would be paid by multiplying their hours by their wage and subtracting 25% for taxes. For example:  
   (pay ‘(Bob 10.0 40))  
   would return  
   300

Section 4: Semantics

1. Define semantics.
2. What is a block and what does it mean for a language to be block-structured?
3. Define each of the following dangers in programming language assignment semantics and explain why they are dangerous:
   1. Aliasing
   2. Dangling References
   3. Garbage
4. Give an example of a piece of code in C++/Java/Python that creates each of the following problems:
   1. Aliasing

Example 1:

Example 2:

* 1. Dangling Reference:

Example 1:

Example 2:

* 1. Garbage:

Example 1:

Example 2:

1. State whether the following variable properties are bound statically or dynamically in C++.
   1. Name
   2. Address
   3. Value
   4. Type
2. Describe the three areas of variable allocation in memory, including hat is stored there, when they are allocated, and if/how they grow at runtime.

(Unallocated)

1

2

3

Static Area (global variables)

Stack (declared variables, local variables)

Heap(dynamically allocated variables): ie, malloc

1. Draw the three different possible semantic interpretations of the statement:  
   a = b;

Storage semantics(location don’t change) ie, C/C++

Assignment by sharing (pointer semantics) ie, SNOBOL, LISP

Assignment by Cloning ie, Java

1. Given the following C++ code:

int x;

void Top()

{ int x = 42;

Bottom();

}

void Bottom()

{ int z = 75;

cout << x << endl;

}

void main()

{ double y = 7.5; x = 15;

Top();

}

* 1. Identify all the local and nonlocal variables in this code.
  2. Show what is displayed under dynamic scoping.
  3. Show what is displayed under static (or lexical) scoping.
  4. Which type of scoping does C++ (and Java) actually use?

1. What causes a stack overflow error in a program? Is the error normally caught at runtime or compile time?
2. Give an example in C++/Java/Python of a piece of code with a scope hole. How does the scope resolution operator address the problem of scope holes?

Ie1,

ie2,

1. Write a variable declaration or assignment statement in C++/Java/Python.
   1. What are the four attributes that are bound to a name associated with a variable?
   2. Which of those attributes are explicitly and implicitly bound in the language that you used?
   3. Which of those attributes are bound statically and dynamically in the language that you used?