1		/ 30
2		/ 30
3		/ 50
4		/ 5
5		/ 15
6		/ 15
7		/20
8		/20
9		/ 15
total		/ 240

UMBC CMSC 331 Final Exam

Name:________________________________ UMBC Username:___________________________

You have two hours to complete this closed book exam. We reserve the right to assign partial credit, and to deduct points for answers that are needlessly wordy or simply wrong.

1. General multiple-choice questions (30)

Circle the **all** of the letters for correct answers for each problem (4 points each).

1.1 A common way to define a programming language's syntax is to use a (a) binary tree; (b) box and pointer diagram; (c) context free grammar; (d) LRRL grammar; (e) BNF.**(C)**

1.2 Attribute grammars are typically used to (a) Handle left-recursion. (b) Handle language features which context-free grammars can not. (c) Prove program correctness. (d) Compile grammars into efficient tables. **(B)**

1.3 The Unix LEX program is used to define the (a) rules for tokenizing the input to a parser; (b) lexical attributes for an attribute grammar; (c) the lexical structure of a language; (d) a bottom up shift-reduce parser.**(B)**

1.4 The Unix YACC program parses input using a (a) recursive descent parser; (b) a top-down parser; (c) an RR(1) parser; (d) a bottom-up shift reduce parser. **(A)**

1.5 The main difference between a sentence and a sentential form is (a) there is no difference; (b) a sentence contains only terminal symbols but a sentential form can contain some non-terminal symbols; (c) sentential forms are a subset of sentences but the converse is not true; (d) sentential forms have no handles but a sentence does. **(A)**

1.6 If class Oof is the parent class of Rab, which of the following is illegal Java syntax: (a) Rab rab = new Oof(); (b) Rab oof = (Rab)(new Oof()); (c) Oof oof = new Rab(); (d) Rab rab = new Rab(); (e) Oof rab = (Rab)(new Oof()); **(A)**

1.7 If class Jello implements interface Wiggles, which of the following is illegal Java syntax: (a) Wiggles w = new Jello(); (b) Jello w = (Jello)(new Jello()); (c) Wiggles w = (Wiggles)(new Jello()); (d) Jello j = new Wiggles(); (e) Jello w = new Jello(); **(D)**

1.8 What is the relation between classes and objects: (a) Objects must be declared before their classes can be used. (b) Objects describe the behavior of their classes. (c) Objects have class. (d) A class is an instantiated object. (e) Classes describe types and objects are the values of those types. **(E)**

1.9 In an attribute grammar, what is the difference between synthesized and inherited attributes: (a) Synthesized attributes are initialized by the scanner. (b) X Synthesized attributes depend only on information below them in the parse tree. (c) Inherited attributes are useful only in object-oriented languages. (d) Synthesized attributes are computed at run time. **(B)**

1.10 LISP macros are primarily used to define: (a) dynamically scoped environments; (b) closures; (c) functions that don't evaluate all of their arguments; (d) reflective programs. **(C)**

2. Grammars (30: 10/10/10)

In the grammars shown below, upper case letters denote non-terminal symbols, lower case letters denote terminal symbols, the start symbol is always S, and ε denotes the empty string. For each grammar: (1) Describe the language generated by the grammar concisely in English. For example: "All strings of zeros and ones which contain at least 5 zeros." And (2) say whether or not the grammar is ambiguous. If the grammar is ambiguous, find a string that demonstrates the ambiguity, and draw two distinct parse trees for this string.

(a) $S \rightarrow aSa \mid bSb \mid cSc \mid a \mid b \mid c \mid \varepsilon$

Strings containg any number of a's b's and c's in any order, followed by the same sequence of a's b's and c's in reverse order. The grammar is not ambiguous.

(b) $S \rightarrow \varepsilon \mid ab \mid ba \mid aSb \mid bSa$

Stings with a prefix consisting of some number of a's and b's in any order followed by a postfix formed by reversing the prefix and changing each a to a b and each b to an a. The grammar is not ambiguous.

(c) $S \rightarrow aSb \mid C \quad C \rightarrow \varepsilon \mid c \mid Cc$

Strings consisting of N a's (N≥0) followed by any number of c's followed by N b's. The grammar is not ambiguous.

3. Java true/false questions. (50)

Mark each assertion as true or false by circling [T][F]. (2 points each)

[T][F]: A char is just a String of length 1. **[F]**

[T][F]: An abstract class cannot be instantiated. **[T]**

[T][F]: Methods declared in an interface are automatically public. **[T]**

[T][F]: Overloading and overriding refer the same thing in Java **[F]**

[T][F]: the statement **new Object[7];** creates an array of null references. **[T]**

[T][F]: A subclass can access private variables in its superclass. **[F]**

[T][F]: A subclass may not access a private member of an ancestor class. **[T]**

[T][F]: Making a class final means that it can not have any new instances. **[F]**

[T][F]: Instance variables defined by a class are always given initial values when an instance of the class is created. **[T]**

[T][F]: Method variables in a method are always given initial values when the method is called on an instance. **[F]**

[T][F]: A method's signature only includes its name, return type and the number and types of its arguments. **[T]**

[T][F]: The values of static final variables in a class can not be changed after initialization. **[T]**

[T][F]: Static methods do not require an instance of the class; they can be accessed through the class name. **[**

[T][F]: A Java try statement can have more than one catch statement. **[T]**

[T][F]: Every try expression must be followed by at least one catch expression. **[T]**

[T][F]: In Java ,Vectors differ from Arrays in that their size is not fixed and can be changed dynamically.**[T]**

[T][F]: Executing an assert statement checks a condition that the programmer expects to be true. **[T]**

[T][F]: If o1 and o2 are declared Object references and o1.equals(o2) is true, then o1==o2 must also be true. **[F]**

[T][F]: Java does not have true multiple inheritance like C++. **[T]**

[T][F]: it would not make sense for two classes to mutually inherit (each inherits from the other) because they'd end up being effectively the same class. **[T]**

[T][F]: Since arrays are objects in Java they inherit all the characteristics of java.lang.Object. **[T]**

[T][F]: To execute code that is declared to throw an Exception, we must either declare ourselves to do the same or enclose that code in a try catch/block. **[T]**

[T][F]: In Java 1.5, a Vector that is not parameterized can store instances of any kind of Object, but they all have to be of the same type. **[F]**

[T][F]: In Java 1.5, autoboxing is a mechanism for automatically creating an instance of a wrapper class for a primitive data type value. **[T]**

[T][F]: Downcasting is generally required when your program takes instances out of a vector or arrayList unless they are parameterized. **[T]**

4. Java class variables (5)

If we construct an object of type Foo, where **class Foo { int x = 5; int y; Foo() { x = 2; } }**, what are the values of x and y in the new object?
x=2 and y=0

5. Constructing s-expressions (15: 5/5/5)

Consider the Lisp data Structure that when printed looks like (((A) B) C)

5.1 Give a LISP expression using only the CONS function that will create this list.

(cons (cons (cons ‘a nil) (cons ‘b nil)) (cons ‘c nil))

5.2 Give a LISP expression using only the LIST function that will create this list.

(list (list (list ‘a) b) ‘c)

5.3 Assuming that we've done **(SETF X '(((A) B) C))** give s-expression using only the functions CAR and CDR and the variable X that would return each of the three symbols in the list.

symbol	*s-expression to return the symbol*
A	**(car (car (car x)))**
B	**(car (cdr (car x)))**
C	**(car (cdr x))**

6. Writing a Lisp function (15: 10/5)

A classic LISP data structure is the Association List or ALIST. An ALIST is a list of items, each of which has a symbol and an arbitrary s-expression. For example:

((AGE 20) (MAJOR "CMSC")(COURSES (CMSC331 CMSC311 BIOL340))

(6a) Write a function ASSOC that takes two arguments, a symbol and an ALIST, and returns the item associated with that symbol (if any) in the ALIST or NIL if there is no item associated with the symbol. E.g.:

```
 > (SETF A '((AGE 20) (MAJOR "CMSC")(COURSES (CMSC331 CMSC311 BIOL340)))
((AGE 20) (MAJOR "CMSC")(COURSES (CMSC331 CMSC311 BIOL340)))
> (ASSOC 'AGE A)
(A 20)
> (ASSOC 'COURSES A)
(COURSES (CMSC331 CMSC311 BIOL340))
> (ASSOC 'HATSIZE A)
NIL
(DEFUN ASSOC (X ALIST)
  (COND ((NULL ALIST) NIL)
        ((EQ X (CAR (CAR ALIST))) (CAR ALIST))
        (T (ASSOC X (CDR ALIST)))))

;; Here's an alternate way to di it using MEMBER

(DEFUN ASSOC (X ALIST)
   (CAR (MEMBER X ALIST :TEST #'(LAMBDA (S AL) (EQ S (CAR AL))))))
```

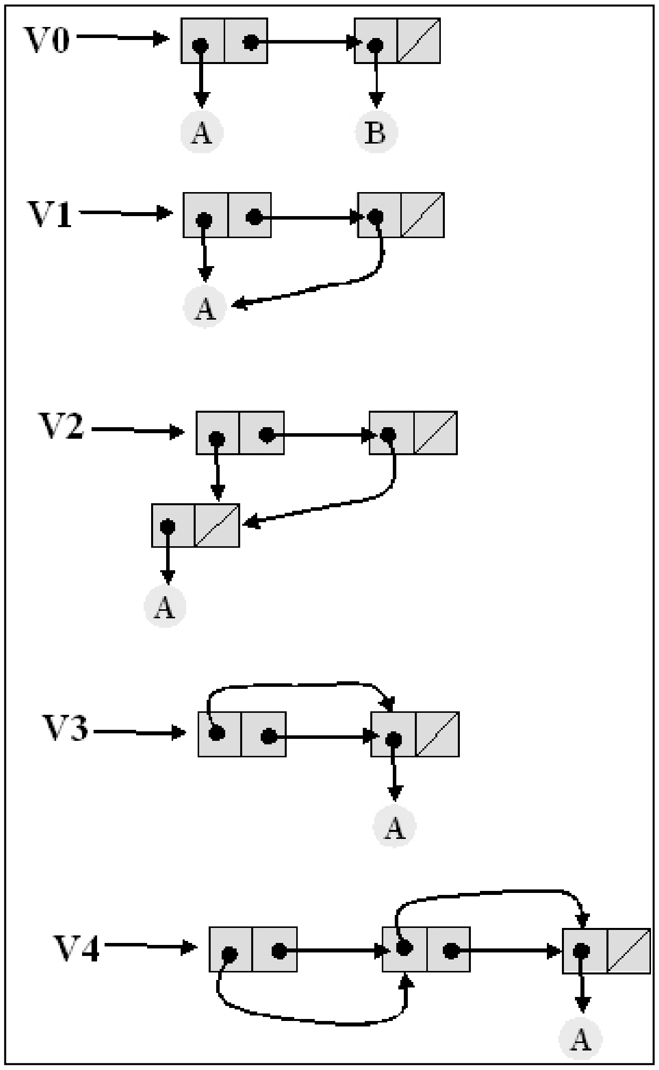
(6b) Why is it a good idea to have this function return the pair (symbol + value) rather then just the second element of the pair (i.e., value)?

If we just return the s-expression associated with a symbol, then getting back NIL from ASSOC would be ambiguous. Was the item not found, or was it found but the associated s-expression was NIL? For example, if AL is ((TALL T) (YOUNG NIL)(HAPPY T)) consider what would be returned by (ASSOC 'YOUNG AL) versus (ASSOC 'RICH AL).

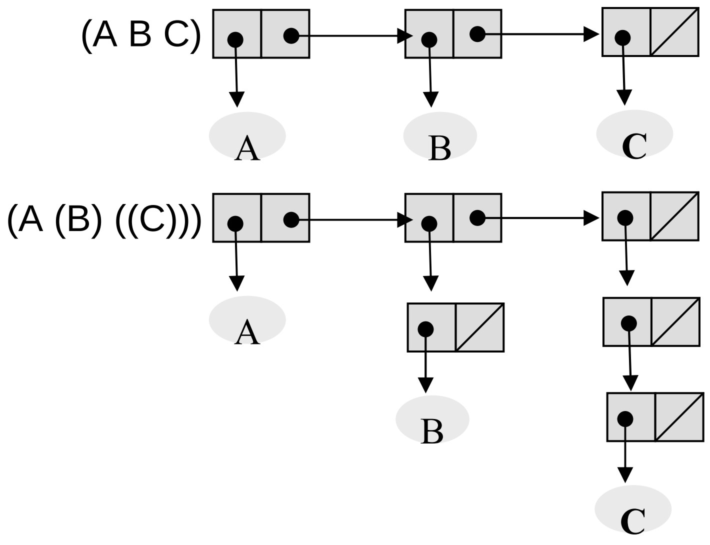
7. List structures (20: 10/5/5)

(7a) Show what would be printed for each of the five list structures to the right.

	What would be printed?
V0	**(A B)**
V1	**(A A)**
V2	**((A) (A))**
V3	**((A) A)**
V4	**(((A) A) (A) A)**



Draw a box and pointer diagram for each of the following lists.
(7b) (A B C) (7c) (A (B) ((C)))



8. Functional programming I (20: 10/10)

```
> (MAXINT NIL)
0
> (MAXINT '(3))
3
> (MAXINT '(3 2 4 9 1 2))
9
```

Consider a function MAXINT that takes a list of positive integers and returns 0 if the list is empty and the largest value in the list if it is not empty. Examples of its behavior are shown in the box to the right.

(8a) Write a straightforward recursive version without using any additional local variables. Hint: define MAXINT to take a list of integers, but have it call a recursive function MAXINT1 which takes two arguments – a list of integers and the largest integer seen so far. Three things to consider here: (1) what initial call to MAXINT1 should MAXINT make; (2) What should MAXINT1 return if the list is empty; (3) what should MAXINT1 return if the list is not empty?
This will get you started:
(DEFUN MAXINT (L) (MAXINT1 …..))
(DEFUN MAXINT1 (…) …)

```
(DEFUN MAXINT (L) (MAXINT1 L 0))

(DEFUN MAXINT1 (L N)
   ;; here is one version of maxint1
   (IF (NULL N) 0 (MAXINT (CDR L) (MAX N (CAR L)))))

(DEFUN MAXINT1 (L N)
  ;; here is a variation on maxint1
  (COND ((NULL L) N)
        ((> (CAR L) N)(MAXINT (CDR L)(CAR L)))
        (T (MAXINT (CDR L) N))))
```

```
(DEFUN REDUCE (L FN ID)
  (IF (NULL L)
      ID
      (FUNCALL
         FN
         (CAR L)
         (REDUCE (CDR L) FN ID))))
```

(8b) Now write a version that uses the function reduce discussed in class and shown to the right. Hint: MAXINT should be a one liner that just calls REDUCE. Consider using the built in function MAX that takes two numbers and returns the larger.

(DEFUN MAXINT1 (L) (REDUCE L #'MAX 0))

9. Functional programming (15: 3/3/4/5)

Another aspect of functional programming is the ability to write functions to create new functions. Consider the function XYZZY defined as follows:

```
    (DEFUN XYZZY(F G)(LAMBDA(X)(FUNCALL F (FUNCALL G X))))
```

9.1 With what type of arguments should xyzzy be call? **Two function**
9.2 What type of thing does xyzzy return?**A function**
9.3 Describe in a sentence what xyzzy does? **Given two unary functions, F and G, XYZZY will return their composition, i.e., a function that applies F to the result of applying G to an argument.**
9.4 What will the following expression return?
(MAPCAR (XYZZY (LAMBDA (X)(+ X 3)) (LAMBDA (X)(* X X))) '(1 2 3 4 5 6))
(4 7 12 19 28 39)