1		/ 5
2		/ 10
3		/ 30
4		/ 50
5		/ 10
6		/ 15
7		/15
8		/15
9		/ 20
10		/15
11		/15
total		/ 200

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. Good luck.

1. Java tricky question (5)

What is printed by the following? Think carefully.

```
int x = 0;
for (int i = 0; i < 5; i++) {
  x = x++;
  System.out.print(x);
  }
```

00000

Comment: Well, we said it was tricky. The expression x++ increments x but evaluates to the old value. So, if x is 0, then the value of x++ *is 0. Evaluating it has the side effect of assigning x to be 1. But it's the value that is used by the assignment statement. So* x=x++ *is essentially a no-op statement. It has no effect. The value of x is not changed.*

2. A Simple Java method (10)

Write a method named **outOfPlace** that when given a nearly-but-not-quite sorted int array returns the number of array elements that are "out of place." An array element is "out of place" if it's larger than the one that immediately follows it.

```
int outOfPlace(int[ ] array) {
  int count = 0;
  for (int i = 0; i < array.length - 1; i++) {
    if (array[i] > array[i + 1]) count++;
   }
  return count;
}
```

3. General multiple-choice questions (30)

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

1.1 Assuming normal Java naming conventions, **Display** could be the name of (a) a class; (b) a variable; (c) a method; (d) an interface; (e) a primitive datatype. **(A,D)**

1.2 Assuming normal Java naming conventions, **display** could be the name of (a) a class; (b) a variable; (c) a method; (d) an interface; (e) a primitive datatype. **(B,C)**

1.3 A method declared in an interface is implicitly (a) private; (b) public; (c) protected; (d) abstract; (e) concrete; (f) none of the above. **(B,D)**

1.4 Which of the following is NOT a valid function call in Javascript? (a) var x=myfunc(); (b) myfunc; (c) x=myfunc(); (d) myfunc(); **(B)**

1.5 In Javascript the expression A ? B : C is the equivalent to: (a) if (A) {B} else {C} ; (b) if (A==B) C ; (c) if (A!=B) C; (d) if (A) {B;C}; (e) none of the above. **(A)**

1.6 Which phrase best describes the variable typing scheme used in Javascript? (a) static typing; (b) dynamic typing; (c) flexible typing; (d) touch typing; (e) typeless; (f) object oriented typing; (g) none of the above. **(B)**

1.7 Which of the following expressions would be interpreted as false when evaluated: (a) NIL; (b) (LIST); (c) (CDR '(100)); (d) '(); (e) none of the above. **(A,B,C,D)**

1.8 Which phrase best describes the variable typing scheme used in LISP? (a) static typing; (b) dynamic typing; (c) flexible typing; (d) touch typing; (e) typeless; (f) object oriented typing; (g) none of the above. **(E)**

1.9 In LISP a lambda expression represents a (a) abstract class; (b) variable type; (c) function; (d) conditional; (e) cons cell. **(C)**

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)**

1.11 Which of the following is not considered a functional programming language? (a) ML; (b) Haskell; (c) Smalltalk; (d) Scheme; (e) Lisp. **(C)**

1.12 In the standard Model-View-Controller design pattern, which component(s) detect and responds to GUI events: (a) Model; (b) View; (c) Controller; (d) none of the above. **(C)**

1.13 In Java, which of these is not part of a method's signature: (a) name; (b) number of parameters; (c) types of parameters; (d) type of return value. **(D)**

1.14 In an object-oriented language, when an object has several methods have the same name but different signatures, we call this (a) inheritance; (b) overriding; (c) overloading; (d) signature clash. **(C)**

1.15 Java 1.5, generic types were introduced to (a) eliminate wrapper classes; (b) reduce the need for downcasting; (c) provide autoboxing; (d) make Java Type safe; (e) none of the above. **(B)**

4. Java true/false questions. (50)

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

[T][F]: You can write a constructor for an abstract class. **[T]**

[T][F]: You can create an instance of an abstract class.. **[F]**

[T][F]: If **AbsClass** is an abstract class, the expression **x instanceof AbsClass** is always false. **[F]**

[T][F]: You can declare a constructor in an interface. **[F]**

[T][F]: You can override an inherited method to make it private. **[F]**

[T][F]: You can override an inherited method and throw additional checked exceptions. **[F]**

[T][F]: You can override an inherited method and throw fewer checked exceptions. **[T]**

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

[T][F]: Java jar files can contain only compiled versions of Java classes (e.g., .class files). **[F]**

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

[T][F]: It is legal to have methods **void foo(String x)** and **int foo(Object x)** in the same class. **[T]**

[T][F]: It is legal to have methods **void foo(String x)** and **int foo(String y)** in the same class. **[F]**

[T][F]: Classes, methods, and instance variables can all be declared **private**. **[T]**

[T][F]: Classes, methods, and instance variables can all be declared **final**. **[T]**

[T][F]: Classes, methods, and instance variables can all be declared **void**. **[F]**

[T][F]: Java vectors differ from arrays in that their size is not fixed and can be changed dynamically. **[T]**

[T][F]: A class can extend only one other class. **[T]**

[T][F]: A class can implement at most one interface. **[F]**

[T][F]: A class can not be implemented by a second class and extended by a third class. **[F]**

[T][F]: Javadoc comments can be generated automatically by the Java compiler or BlueJ IDE. **[F]**

[T][F]: A member class can access **private** instance variables of the enclosing class. **[T]**

[T][F]: ~~To~~ A local inner class can access local variables of the enclosing method. **[F]**

[T][F]: Swing was the original Java GUI toolkit and the AWT was added in Java 2. **[F]**

[T][F]: A Java file must define exactly one non-anonymous class. **[F]**

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

[T][F]: False (F) is your answer to this question. **[?]** *comment: we won't score this question, of course. There is no consistent way to answer this question. It's a variation on the famous liar's paradox.*

5. Java loops (10: 5/5)

Show what is printed for each of these Java statements.

(a) `int i = 4; while (i < 10) { i++; System.out.print(i + " "); }`

5 6 7 8 9 10

(b) `int i = 4; do { System.out.print(i + " "); i++; } while (i < 10);`

4 5 6 7 8 9

6. Classy people (15: 5/5/5)

Write a Java class **Person** with instance variables **String firstName** and **String lastName**. (a) Provide a constructor that takes two string arguments providing values for the first and last names. (b) Override the methods **toString** to return a person's full name as a string. (c) Override the method **equals** to return true if two Person objects had the same first and last names.

```
public class Person
{
        public String firstName;
        public String lastName;

        public Person(String fn, String ln)
        {
                firstName = fn;
                lastName = ln;
        }

        public String toString()
        {
                return firstName + " " + lastName;
        }

        public boolean equals(Object o)    // the method should match for any object!
        {
           if (o==null) {                  // your method should work for null!
              return false;
             }
           Person p = (Person) o;
           return (p.firstName.equals(firstName) &&
                     p.lastName.equals(lastName));
        }
}
```

7. 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 ‘A (cons (cons ‘B NIL) (cons ‘C NIL)))
```

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

```
(list ‘A (list ‘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 X)
B	(car (car (cdr X)))
C	(car (cdr (cdr X)))

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

```
> (SUMLIST NIL)
0
> (SUMLIST ‘(100))
100
> (SUMLIST ‘(1 2 3 4 5))
15
```

This question asks you to write two versions of the Lisp function SUMLIST that takes a list of numbers and returns their sum, as shown in the box to the right. Assume that your function is always passed a valid argument.

(a) Write SUMLIST as a simple recursive function.
Comment: here are two variations:

```
(defun sumlist (l)  (if (null l) 0 (+ (car l) (sumlist (cdr l)))))
(defun sumlist (l)  (cond ((null l) 0) (T (+ (car l) (sumlist (cdr l))))))
```

(b) Write SUMLIST without using recursion or iteration, but using either APPLY or FUNCALL

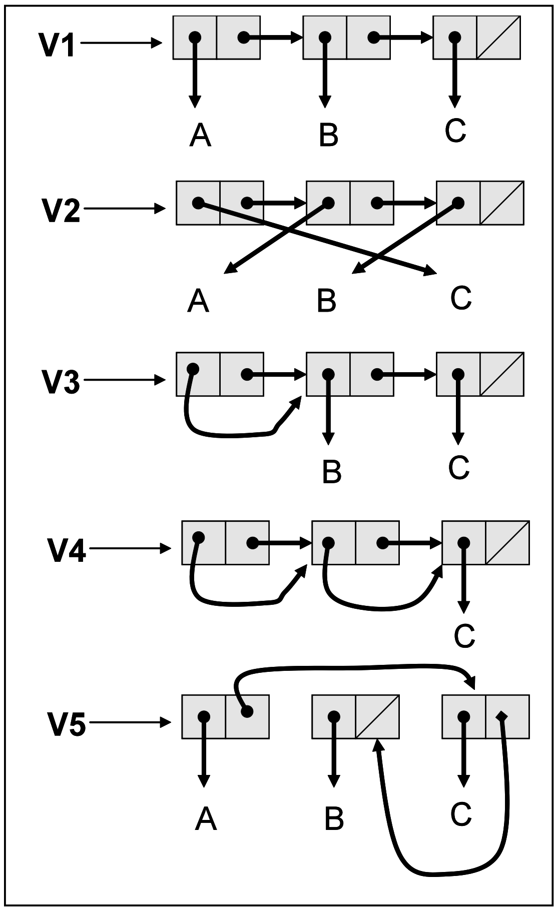
Comment: here are two variations:

```
(defun sumlist (l) (apply #’+ l))
(defun sumlist (l) (funcall #’+ l))
```

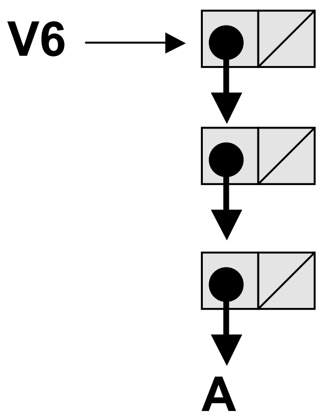
9. List structures (20: 10/5/5)

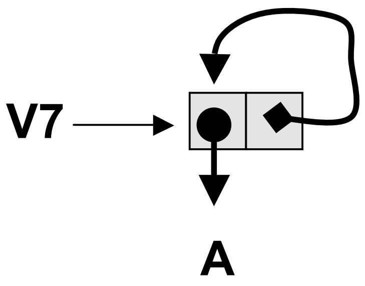
(7a) Show what would be printed for each of the five list structures to the right.

	What would be printed?
V1	**(A B C)**
V2	**(C A B)**
V3	**((B C) B C)**
V4	**(((C) C) (C) C)**
V5	**(A C B)**



(7b) Draw a box and pointer diagram for V6: (((A)))
(7c) Draw a finite box and pointer diagram for V7: (A A A …..); i.e., an infinitely long list of As





10. Functional programming I (15: 5/5/5)

```
> (defun filter (F L)
    ;; returns elements of  L for which F is true.
    (cond ((null L) nil)
          ((funcall F (car L))
           (cons (car L)  (filter F (cdr L))))
          (t  (filter F (cdr L)))))
filter
> (filter #'evenp '(1 2 3 4 5 6))
(2 4 6)
```

Recall the filter function shown to the right. It takes two arguments: F, a single argument predicate function and a list L and returns a list of L's elements for which F is true. Suppose the variable **peeps** is bound to a list of sub-lists representing a person's name, age, sex and major, as in:

```
(setf peeps '( (john 19 male cmsc)  (mary 23 female cmpe)  (sue 20 female ifsm)
               (bill 21 male cmsc)  (alice 18 female cmsc)  …  )
```

Using filter and anonymous functions defined as lambda expressions, give expressions that return a list of people in **peeps** that:

Comment: we need to filter the list with a function that takes a person structure (e.g., (john 19 male cmsc) and returns true it it matches the desired constraints and false otherwise. It's just a matter of writing a boolean condition on parts of the person structure. There are several ways to write each of the expressions below.

(a) Are aged 21 or older.

```
(filter (lambda (p) (> (car (cdr p)) 20) peeps)
```

(b) Are women aged 21 or older

```
(filter (lambda (p) (and  (> (car (cdr p)) 20) (equal (caddr p) 'female) )) peeps)
```

(c) are women CMPE or CMSC majors older than 18

```
(filter (lambda (p) (and  (> (car (cdr p)) 18)
                           (eq (caddr p) 'female)
                          (member (cadddr p) '(cmsc cmpe))))
        peeps)
```

11. Functional programming II (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:

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

9.1 With what type of arguments should xyzzy be called?

A function of one argument

9.2 What type of thing does xyzzy return?

A function of one argument

9.3 Describe in a sentence what xyzzy does?

Given a unary function F, XYZZY will return a new function which is equal to F composed with itself.

9.4 What will this expression return: (MAPCAR (XYZZY (LAMBDA (X) (* X X))) '(1 2 3 4 5 6))

Comment: (lambda (x) (* x x)) *is just an anonymous function that squares its numerical argument. So XYZZY applied to this returns a function that takes the square of the square of its argument, i.e., it raises its argument to the fourth power. So, mapping this down the list of the first six integers produces the following answer.*

(1 16 81 256 625 1296)