CMSC 331 Final Exam
Section 0201 – December 18, 2000

Name: ___Answers_____
Student ID#:_____

You will 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.

0. Logic (0)

The multiple-choice question below has only one correct answer. Which one is correct?

(a) Answer (a) or (b)
(b) Answer (b) or (c)
(c) Answer (c)

0	00/
1	20/
2	10/
3	15/
4	40/
5	10/
6	10/
7	20/
8	20/
9	10/
10	15/
11	15/
12	10/
13	10/
14	15/
15	20/
*	240/

1. General multiple-choice questions (20)

1. Which kind of variable has the longest lifetime?
   (a) An implicitly heap-dynamic variable
   (b) A static variable
   (c) A stack-dynamic variable
   (d) A flexible variable
   (e) A fixed-length variable
   (f) A lexically scoped variable

2. Which of the following is true of aliases?
   (a) An alias changes the name of something
   (b) An alias protects an existing value from being overwritten
   (c) An alias provides an alternative way of accessing something
   (d) An alias allows type inference
   (e) Aliases should be avoided if at all possible
   (f) Aliases support name equivalence of record types

3. What happens in an assignment such as ``x := y''?
   (a) The address of x is modified to be the address of y
   (b) The address of y is modified to be the address of x
   (c) The object bound to y is copied and bound to x, and any previous binding of x to an object is lost
   (d) x and y become aliases

4. Which of the following is the strictest, as applied to record types?
   (a) name equivalence
   (b) structural equivalence
   (c) subtype agreement
   (d) assignment compatibility

5. Which of the following pairs of languages have type inferencing?
   (a) ML and Haskell
   (b) APL and SNOBOL
   (c) Fortran and PL/I
   (d) LISP and Prolog

6. Which of the following is true of l-values and r-vlaues
   (a) An l-value is a logical value, and an r -value is a real value.
   (b) l-values are always to the left of r-values
   (c) An l-value refers to a variable's location while an r-value to its current value
   (d) L-values are local and r-values are relative.

7. Which of the following situations will create a ``dangling pointer"?
   (a) setting an arbitrary pointer variable to ``null"
   (b) setting the CDR part of a CONS cell somewhere in the middle of a list to ``null"
   (c) freeing a block to which there still exists a live pointer
   (d) freeing a block which still contains pointers to other existing blocks

8. In Pascal, one problem with union types is that
   (a) Sets are implemented as arrays and are thus inefficient.
   (b) Every union type has to be defined as an enumeration type.
   (c) Pascal does not require a type discriminator field, so that it is possible to violate type safety.
   (d) Type inferencing is not supported across union types.

9. Which of these languages does not have a primitive data type for a character string:
   (a) Common Lisp (b) Pascal (c) Ada **(d) Java**

10. What distinguishes a purely ``functional" programming language from an ``imperative" one?
   (a) There are no variables and hence no assignment operation in a purely functional language
   (b) A purely functional language lacks the ``go to" statement, but an imperative language always has such a command
   (c) All subprograms must be declared with the keyword function in a purely functiona l language
   (d) There is no real difference, only a difference in the recommended coding style

2. Advantages and disadvantages of virtual machines? (10)
Java and other programming languages have taken the approach of executing on a virtual machine, as opposed to compiling to native machine code. Explain one advantage and one disadvantage of the virtual machine approach. ‘

Some advantages are: greater portability, greater security and resource control by building restrictions into the VM, more uniform execution environment (e.g., AWT provides a single look and feel), bytecode programs will be smaller than machine language programs, so transmitting them across a network is quicker. Some disadvantages are: less efficient execution than in compiling to machine code, the uniform execution environment is likely to be the least common denominator.

3. Relating syntax concepts (15)
Briefly explain the relationships between the following syntax terms: parse tree, derivation, grammar, language, and sentence.

A language consists of a set of legal sentences. A grammar specifies which sentences care legal and thus in the language. Grammars also associate each sentence with one or more derivations, each of which corresponds to a parse tree. A parse tree is a tree structure which shows one of the structure assigned to the sentence by the grammar.

4. Java true/false (40)

1. In Java array indices start at 0. **True**
2. A Java class cannot contain both class and instance methods. **False**
3. To allow Java's automatic garbage collection mechanism reclaim the memory used by an object, one can assign the keyword null to an object reference. **True**
4. Since arrays are objects in Java they inherit all the characteristics of java.lang.Object. **true**
5. In Java an array's length can chan ge at any time. **False**
6. In Java, a string is not a primitive data type but rather is defined as an object. **true**
7. There are three categories of variables in Java: class variables, instance variables, and local variables. **True**
8. A Java array declared for an object type is really an array of object references, so creating the array, with new, simply creates empty references with the elements of the array are all set to null. **true**
9. Java's interface mechanism provides all of the benefits of multiple inheritance but do es so more efficiently than C++. **false**
10. Java's AWT gives the programmer more control over the look and feel of a graphical user interface than does SWING. **false**
11. All Java objects are descended from Java.Object. **true**
12. At most one public class may be defined by any one Java source file. **true**
13. If a public class appears in a Java source file, the name of the file and the name of the class must be the same. **True**
14. The value of a static variable cannot be changed once it has been initialized. **false**
15. A Java class can have only one superclass but can extend any positive number of interfaces. **true**
16. All objects of the same class share the same fields (if you change the field associated with one object all objects will see the same change.) **False**
17. System is a primitive data type in Java as shown by the use of System.out.println to output information. **False**
18. If a member is declared to be private, it can still be accessed by any class defined in the same package. **False**
19. The model-view-controller pattern most of the graphical ro utines are in the model. **false**
20. A Java class cannot contain both class and instance methods. **False**

5. Evaluating Java (10)

What gets printed by the following Java code fragment?

```
int  i = 3, j = 3, k;
k = ++i + ++j + 2 ;
System.out.println( i + "  " + j  + "  " + k );
i = j = 3;
k = i++ + j++ + 2 ;
System.out.println( i + "  " + j + "  " + k );
i = 2;  j=3 ;k = 4;
k *= i - j;
System.out.println( k );
```

4 4 10
4 4 8
-4

6. Short Java answers (10)

1. Suppose that takeFinal is a method of a class named Stud ent. The takeFinal method takes no arguments. Write an example of calling the method takeFinal for an instance of Student named aStudent. **aStudent.takeFinal();**
2. What happens when an instance of an object is no longer referenced by any variable, array entry , or other collection **o**f objects in Java? **It gets garbage collected.**
3. In declaration *public static final int SIZE = 100;* which of the modifiers makes SIZE a class variable? **Static**
4. In Java, new classes can be defined by extending existing classes. This is an example of: (a) encapsulation, (b) interface (c) polymorphism (d) inheritance (e) unification (f) objectification. **(d) inheritance**
5. Consider the following fragment of code:

```
int field1 = 100;
int field2 = 200;
field2 = field1;
field1 = filed2;      // line 4
```

When the code has executed through the line with the comment "line 4", what are the values of field1 and field2? **100 and 100**

7. Lisp true/false (20)

1. The Lisp function EQ is a test of pointers. **True**
2. In Lisp NIL is considered both an atom and a lis t. **True**
3. If L is bound to a non-nil list, then (eq L (reverse (reverse L))) always evaluates to T. **false**
4. In Lisp, the integer 0 represents logical *false* and any other integer represents logical *true*. **False**
5. The only way to introduce new user-defined specia l forms is by using macros. **True**
6. The *lambda* special form can be used to define "anonymous" functions in Lisp. **True**
7. A special form is a function that does not evaluate all of its arguments or evaluates them in a non - standard way. **True**
8. Lisp's macro facilit y makes it easy to extend the language. **True.**
9. Every s-expression in Lisp returns a value when evaluated. **True**
10. Lisp does not allow you to assign a variable more than once. **False**

8. Lisp lists (20)

After evaluating *(setq alpha '(1 2 3) beta '(a b c) gamma '(do re mi) empty '())*, what would be the value of each of the following Lisp expressions.

1. (cons alpha beta) **((1 2 3) a b c)**
2. (cons (car alpha) beta) **(1 a b c)**
3. (cons (car gamma) (cons (car beta) (cons (car alpha) empty))) **(do a 1)**
4. (cons (car alpha) (cons (cadr beta) (cddr gamma))) **(1 b mi)**
5. (cons empty empty) **(nil)**

Using **only** the bindings alpha, beta, gamma, and empty as well as the procedures cons, car, and cdr, construct Lisp expressions with the following values.

6. ((a) (b) (c))

```
(cons (cons (car beta) empty)
      (cons (cons (car (cdr beta)) empty)
            (cons (car (cdr (cdr beta))) empty)
                  empty)))
```

7. (3 re c)

```
(cons (car (cdr (cdr alpha)))
      (cons (car (cdr gamma))
            (cdr (cdr beta))))
```

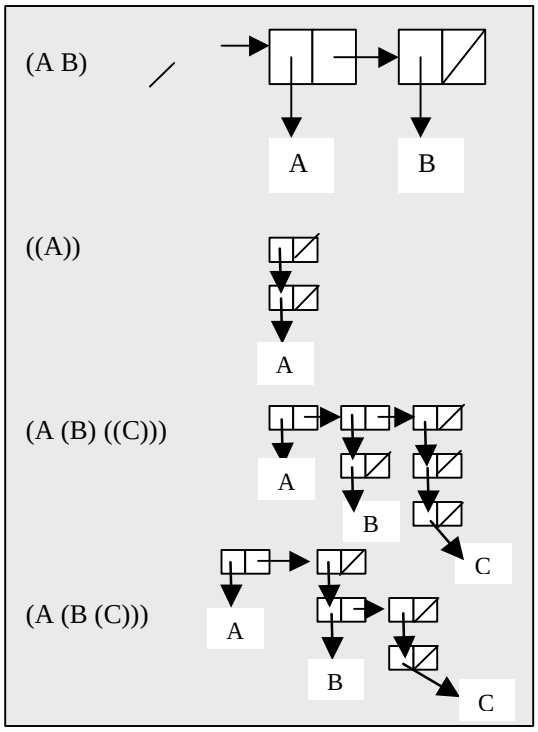
8. ((1 2 3) (a b c) (do re mi))

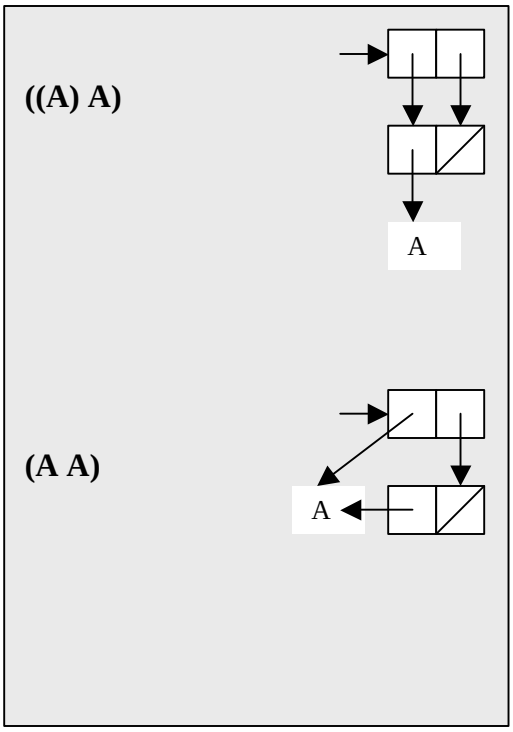
```
(cons alpha
      (cons beta
            (cons gamma empty)))
```

9. ((2 3) (b c) (re mi))
(cons (cdr alpha)
(cons (cdr beta)
(cons (cdr gamma) empty)))
10. (3 2 1)
(cons (car (cdr (cdr alpha)))
(cons (car (cdr alpha))
(cons (car alpha) empth)))

9. More Lisp lists (10)

Complete the following table to show how various Lisp lists would be represented using the standard box-and-pointer diagramming technique. We've done the first one as an example and left the next five for you.





10. Lisp countAtoms function (15)

Write a recursive Lisp function countAtoms that takes a list and returns the number on non-nil atoms in the list. For example:

```
> (countAtoms '((a b)((c)) (((e))) e))
5
> (countAtoms '(((a))))
1
> (countAtoms '())
0
> (countAtoms '(a nil (b) (c nil nil)((nil))))
3
```

```
(defun countAtoms (l)
    (cond ((null l) 0)
          ((atom l) 1)
```

```
(t (+ (countAtoms (car l))
      (countatoms (cdr l))))))
```

11. Lisp butlast function (15)

The lisp function *butlast* takes a list L and returns a new list with the same elements of L except for the last one. For example:

```
> (butlast '(a b c d))
(a b c)
> (butlast '((a b) (c d)))
 ((a b))
> (butlast '(a))
nil
> (butlast nil)
nil
```

Write two version of butlast: (a) one which only uses the function cond, if, car, cdr, and cons. And (b) one that does not use recursion and uses the built-in lisp function *reverse*.

```
(defun butlast (l)
  (cond ((null l) nil)
        ((null (cdr l)) nil)
        (t (cons (car l)
                 (butlast (cdr l))))))
```

```
(defun butlast (l) (reverse (cdr (reverse l))))
```

12. Unification in Prolog (10)

For each of the following pairs of Prolog terms, say whether or not the unification would succeed and, if it does, the variable bindings that would result.. Assume that the same variable na me in two different terms in a given pair represents the same variable. We've done the first one for you as an example.

TERM1	TERM2	Unify?	Bindings
foo(bar,X)	foo(Y,mumble)	yes	Y=bar, X=mumble
X	loves(john,mary).	**Yes**	**X=loves(john,mary)**
p(X, q(Y), r(Z))	p(r(a), q(X), r(b))	**Yes**	**X=Y=r(a), Z=b**
loves(john, X)	loves(Y,Y)	**Yes**	**X=Y=john**
and(father(P1, P2), brother(P1, P3))	and(father(X, john), brother(john, john))	**Yes**	**X=P1=P2=P3=john**
foo(X)	X	**Yes**	**X=f(f(f(f(f(…**
[X, Y | Z]	[a, b, c, d, e, f, g]	**Yes**	**X=a,Y=b,Z=[c,d,e,f,g]**
[ ]	X	**yes**	**X=[ ]**
[X, Y]	[a, b | Z]	**Yes**	**X=a, Y=b, Z=[ ]**
A + B - C	'+'(1, '-'(2, 3))	**Yes**	**A=1, B=2, C=3**
2+3	3+2	**No**	

13. Prolog predicate hasDuplicates (10)

Write a predicate hasDuplicates/1 that take s a single argument that is a list returns true if the list contains multiple occurrences of some top-level element. You can assume that the predicate is always called with an instantiated argument. The hasDuplicates/1 predicate should work like this:

```
?- hasDuplicates([a,b,a,c]).
yes.
?- hasDuplicates([]).
No.
?- hasDuplicates([a,b,[a],c]).
No.
```

Hint: try defining and using the auxiliary predicate isIn/2 that is a variation of member/2:

```
isIn(X,[X|_]).
isIn(X,[_|Tail]) : - isIn(X,Tail).
```

```
hasDuplicates([Head|Tail]) :- isIn(Head,Tail).
hasDuplicates(([_|Tail]) :- hasDuplicates(Tail).

isIn(X,[X|_]).
isIn(X,[_|Tail]) :- isIn(X,Tail).
```

14. Prolog mystery predicate (15)

Assuming append has the usual definition, consider the following mystery predicate:

```
p([ ],[ ]).
p([A|B], C)  :-  p(B, D),  append(D, [A], C).
```

(a) [10] Describe what would happen for each of the following queries in terms of the kinds of errors that would result or the solutions(s) that would be found. If more than one solution would be found, characterize them all.

?? p([a,b,c,d,e],X).

?? **If the first arg is instantiated and the second a variable X, then X is the reverse of the first argument. There is only one solution**

```
| ?- p([a,b,c,d,e],X).
X = [e,d,c,b,a] ? ;
no
```

?? p(X,[a,b,c,d,e])

?? **if the first argument is an uninstantiated variable and the second an instantiated list, then the first solution found is one in which the first argument is unified with the reverse of the second. An attempt to find a second solution does not terminate.**

```
| ?- p(X,[a,b,c,d,e]).
X = [e,d,c,b,a] ? ;
Prolog interruption (h for help)? a
{Execution aborted}
```

?? p(X,Y).

?? **If both arguments are uninstantiated variables, then there are infinitely many solutions found, corresponding to pairs of lists of increasing lengths (0, 1, 2 , …) in which one is the reverse of the other.**

```
| ?- p(X,Y).
X = [],
Y = [] ? ;
X = [_A],
```

Y = [_A] ? ;
X = [_A,_B],
Y = [_B,_A] ? ;
X = [_A,_B,_C],
Y = [_C,_B,_A] ? ;
X = [_A,_B,_C,_D],
Y = [_D,_C,_B,_A] ? ;
X = [_A,_B,_C,_D,_E],
Y = [_E,_D,_C,_B,_A] ? …

(b) [5] Describe what this predicate does in a simple sentence.

P(X,Y) is true if X and Y are lists and X is the reverse of Y. The first argument should be instantiated to insure that it terminated.

15. Prolog true/false (20)

(a) Prolog is a strongly typed language. **false**
(b) The scope of a variable in Prolog is a single clause (i.e., a fact or rule) or a single query. **True**
(c) Unification is transitive (i.e., assuming that t1, t2 and t3 are arbitrary Prolog terms, if t1 unifies with t2 and t2 unifies with t3 then t1 must unify with t3. **false**
(d) The lack of explicit memory allocation and deallocation functions in Prolog is evidence that it performs automatic memory management, e.g. using garbage collection. **true**
(e) One of Prolog strengths is its use of fuzzy logic. **False**
(f) Terms are to Prolog as s-expressions are to Lisp. **true**
(g) The empty list in Prolog is represented by the atomic symbol NIL. **False**
(h) A variable in Prolog must start with either an upper -case letter or an underscore (_). **true**
(i) All local variables must be declared before th ey are used in Prolog. **False**
(j) A Prolog variable can only be assigned to a value once. **true**