0		/ 0
1		/ 20
2		/ 25
3		/ 20
4		/ 25
5		/ 20
6		/40
7		/40
8		/ 30
9		/ 10
10		/ 30
TOT		/ 260

UMBC CMSC 331 Final Exam
Section 0101 – December 17, 2002

Name: _____

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 or simply wrong.

0. Burning rope (0)

There are two lengths of rope each of which can burn in exactly one hour. They are not necessarily of the same length or width as each other nor are they of uniform width, thus burning half of the rope will not necessarily take 1/2 hour. By burning the ropes, how do you measure exactly 45 minutes worth of time?

.

1. General multiple-choice questions (20)

1.1 Most modern programming languages use the following schemes for their variable scoping. (a) lexical scoping. (b) dynamic scoping. (c) both lexical and dynamic scoping. (e) scope inferencing.

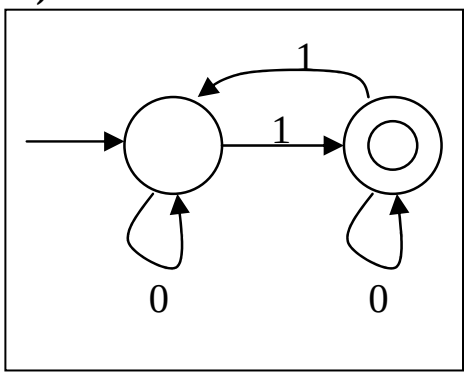
1.2 What happens in an assignment such as ``x := y"? (a) The address of x is modified to be the address of y. (b) The address of y is modified to be the address of x. (c) The object bound to y is copied and bound to x, and any previous binding of x to an object is lost. (d) x and y become aliases. (e) the contents of y are copied into the storage allocated to x.

1.3 Which of the following is true of l-values and r-values? (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. (e) l-values are static and final whereas r-values are dynamic and extensible. .

1.4 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 functional language. (d) There is no real difference, only a difference in the recommended coding style.

1.5 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.

2. Finite state machines and regular expressions (20)



Answer the following questions about the Finite State Machine or FSM (also known as a Deterministic Finite Automaton or DFA) depicted to the right.

2.1 Does this FSM have an initial state and if so which one – left or right? (2)

2.2 Does this FSM have an accepting state and if so which one – left or right? (2)

2.3 Describe in a sentence or two the language that it defines. (5)

2.4 Give a regular expression that defines the same language. Use the following notation for operators: + for one or more repetitions, * for zero or more repetitions, ? for optional expressions, | to separate alternatives. Parentheses can be used for scope. (5) :

2.5 Does every FSM have a regular expression that defines the same language? (2)

2.6 Does every regular expression have a FSM that defines the same language? (2)

2.7 Is it possible for a FSM to define an infinite language? (2)

3. Context free grammars (20)

Consider the following context free grammar written using a BNF notation.

S ::= a B
S ::= b A B
A ::= a
A ::= A A b
B ::= b
B ::= b S
B ::= a B B

3.1 Which symbols are the non-terminal symbols? (2)

3.2 Which symbols are the terminal symbols? (2)

3.3 Is the grammar non-recursive, left-recursive, right-recursive, of both left-and-right recursive? (2)

3.4 Draw a parse tree for the string *baababb* (14)

4. Java true/false questions. (25)

Mark each assertion as true or false by circling [T] or [F].

[T] or [F] : The constructor of a class must not have a return type. [T] or [F] : A class that is abstract may not be instantiated.

[T] or [F] : The value of a static Java variable cannot be changed once it has been initialized.

[T] or [F] : A static variable indicates there is only one copy of that variable.

[T] or [F] : A method defined as private indicates that it's accessible to all classes in the same package.

[T] or [F] : For each try block there must be at least one catch block defined.

[T] or [F] : A try block may be followed by any number of finally blocks.

[T] or [F] : A try block must be followed by at least one finally or catch block.

[T] or [F] : If both catch and finally blocks are defined, catch block must precede the finally block.

[T] or [F] : Arrays in Java are essentially objects.

[T] or [F] : It's not possible to assign one array to another. Individual elements of array can however be assigned.

[T] or [F] : Array elements are indexed from 1 to size of array. [T] or [F] : If a method tries to access an array element beyond its range, a compile warning is generated.

[T] or [F] : Each Java file must have exactly one package statement to specify where the class is stored.

[T] or [F] : If a Java file has both import and package statements, the import must precede the package.

[T] or [F] : A Java file must have at least one class definition.

[T] or [F] : If a Java file has a package statement, it must be the first statement except for possible comments.

[T] or [F] : The Java Virtual Machine compiles Java byte code into machine language instructions.

[T] or [F] : Java does not have true multiple inheritance like C++. [T] or [F] : Java applets, but not Java applications, are run in a "sandbox" that limits the allowed operations.

[T] or [F] : A Java jar file contains Java source code, a manifest, and documentation.

[T] or [F] : As the toString method is defined in the Object class,
System.out.println can be used to print any object.

[T] or [F] : In Java an array's length can change at any time.

[T] or [F] : A Java string is not a primitive data type but rather is defined as an object.

[T] or [F] : A Java class can have only one superclass but can extend any positive number of interfaces.

5. What gets created? (10)

Circle all of the correct answers for each problem. There may be problems for which all of the answers are correct. If all the answers are correct for a specific problem, you should circle all of the choices. There may be problems for which there are no correct answers. If there are no correct answers for a specific problem, you should not circle any of the choices.

1. The Java statement `public class LinkedListStack {}` creates:
   (a) new class; (b) new object; (c) new reference variable; (d) new container to hold objects

2. The Java statement `Vector elements = new Vector();` creates:
   (a) new class; (b) new object; (c) new reference variable; (d) new container to hold objects

3. The Java statement `Vector[] elements = null;` creates:
   (a) new class; (b) new object; (c) new reference variable; (d) new container to hold objects

4. The Java statement `Object element = new Object();` creates:
   (a) new class; (b) new object; (c) new reference variable; (d) new container to hold objects

5. The Java statement `Object element = new Vector();` creates:
   (a) new class; (b) new object; (c) new reference variable; (d) new container to hold objects

6. Short Java answers and multiple choice (40)

Circle all of the correct answers for multiple choice questions.

6.1 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.

6.2 What is a Java expression that can be used to represent number of elements in an array named arrayOne ?

6.3 To represent characters, Java does not use the venerable ASCII but instead uses UNICODE. Briefly describe the difference between ASCII and UNICODE and give two motivations for moving to UNICODE.

6.4 What keyword is used to make a variable belong to a class, rather than being defined for each instance of the class? (a) static (b) final (c) abstract (d) native (e) volatile (f) transient.

6.5 The default layout manager for a Frame is ... (a) FlowLayout (b) BorderLayout (c)GridLayout (d) GridBagLayout (e) CardLayout

6.6 Assume that class A extends class B, which extends class C. Also all the three classes implement the method test(). How can a method in a class A invoke the test() method defined in class C (without creating a new instance of class C). Select the one correct answer. (a) test(); (b) super.test(); (c) super.super.test(); (d) ::test(); (e) C.test(); (f) It is not possible to invoke test() method defined in C from a method in A.

6.7 Consider that Parent and Child classes are defined in two different files as below. What will be output if you try to compile and run above program? (a) It will not compile. (b) It will compile successfully and

print "Parent" and then "Child." (c) It will compile successfully and print "Child" and then "Parent." (d) It will compile successfully, but not run.

```
class Parent{  public Parent(){ System.out.println("Parent");  }   }
class Child extends Parent{
   public Child(int x){ System.out.println("Child"); }
   public static void main(String [] args){ Child c=new Child(10); }
  }
```

6.8 Consider following code. How can you explicitly create an instance of InnerClass? (a) InnerClass i=InnerClass(); (b) InnerClass i=OuterClass.InnerClass(); (c) InnerClass i=new OuterClass ().new InnerClass(); (d) OuterClass.InnerClass i=new OuterClass.InnerClass();

```
public class OuterClass{
   class InnerClass{  }
   public void innerClassDemo(){
      //Explicit instance of InnerClass
   }}
```

6.9 Consider the following code. If you try to compile and run above code what will be result? (a) "Running thread1" following "Running thread2", (b) "Running thread2" following "Running thread1", (c) It will not compile because in Thread1 and Thread2 start() is not defined. , (d) It will not compile because constructor invoked to create Thread2 with arguments (Thread1) is not defined.

```
/** File Thread1.java */
class Thread1 implements Runnable{
  public void run(){ System.out.println("Running Thread1");  }
}  /** End of file Thread1.java */

/** Thread2.java */
class Thread2 extends Thread{
  public void run(){  System.out.println("Running Thread2");  }
  public static void main(String [] args){
     Thread1 t1= new Thread1();
     Thread t2=new Thread2(t1);
     t1.start();
     t2.start();
    }
}  /** End of Thread2.java*/
```

6.10 Consider that class Employee and Salesman are in different file called Employee.java and Salesman.java. What will be result if you try to compile and run above code? (a) Compiler error reported, "Type mismatch: Cannot convert from Salesman to Employee.", (b) It compile successfully and outputs 1000., (c) It compiles successfully and outputs 1100., (d) None of the above

```
/** Employee.java file*/
public class Employee{
int salary=1000;
public int getSalary(){return salary;}
} /**  End of Employee.java file*/

/** Salesman.java file*/
```

```
public class Salesman extends Employee{
int commission =100;
public int getSalary(){return salary+commission;}
public static void main(String [] args){
  Salesman sm = new Salesman();
  Employee em = sm;
  System.out.println(em.getSalary());
  }
} /**  End of Salesman.java file*/
```

7. List processing (20)

Assume that the following has been typed into the Lisp listener:

```
> (setf L1 `(a b)   L2 `(c d)  L3 `(e f))
(e f)
> (defun foo (l)   (if (atom l) nil (append (foo (cdr l) (cons (car l) nil)))))
foo
> (defun bar (one two)
    (setf (car one) (car two))
    (setf (cdr one) (cdr two))
    one)
bar
```

Show what would be printed as the value of each of the following expressions.

```
> (cons l1 l2)



> (append l1 l2)



> (foo L1)



> (cons (car l1)
        (cons (car (cdr l2))
              (cdr (cdr l3))))



> (list (car l1) (car l2))



> (cons (car l1) (car l2))
```

> (setf L4 (list L1 (cdr L2) (foo L3)))

> (list l1 l2 l3)

> (bar l1 l2)

> (list l1 l2)

> (setf (cdr l3) l3)

8. Lisp DNA Matching (15)

You've been hired by Genomagic, a new bioinformatics company which does all of its development in Lisp (there are such companies, btw). Genomagic choose Lisp because a Lisp list is a natural way to represent a DAN, and amino acid sequences. One way to determine a long DNA sequence is to break it into shorter pieces, find the sequences of the pieces, and then re-assemble the pieces based on areas where they overlap with the same codes. This problem asks you to write a few simple functions to do this.

8.1 Complete the definition for the function PREFIX. It takes two arguments which are assumed to be lists of atoms and returns true of the first one is a prefix of the second. For example

```
(prefix '(a b) '(a b c d)) ==>  T
(prefix NIL '(x y z)) => T
(prefix '(a b) '(a a b c d)) => NIL
```

```
(defun prefix (one two)  (cond ...))
```

8.2 Using prefix as a sub-routine, write a recursive function overlap that takes two sequences and returns the length of the common overlapping region (e.g., a tail of the first sequence that is equal to an initial part of the second). For example:

(overlap ‘(c a t g) ‘(t g c a c)) => 2
(overlap ‘(t c t a c t) ‘(a c g t)) => 0
(overlap ‘(c a c t g) ‘(c a c t g)) => 5

(defun overlap (one two) (cond …))

8.3 Using prefix as a sub-routine, write a function DNASPLICE that takes two lists representing DNA sequences and slices them together, removing any common overlapping sequence. That is, if some tail of the first list is equal to some prefix of the second list, remove that tail from the concatenation. For example:

(dnasplice '(a t a) '(c t g)) => '(a t a c t g)
(dnasplice '(c a t a) '(t a c g)) => (c a t a c g)
(dnasplice '(a a a g t g c) '(t g c g t g)) => (a a a g t g c g t g)
(dnasplice NIL '(a t)) => (a t)

9. Lambda expressions (10)

To what do the following expressions evaluate?

9.1 ((lambda (x y z) (* x (+ y z))) 3 5 7)

9.2 ((lambda (x) (* x x)) ((lambda (x) (+ x x)) ((lambda nil 3))))

10. Functional programming (15)

10.1 One aspect of function programming is the use of functions which take other functions as arguments. The Lisp mapcar function is a classic example. What do the following expressions return?

(a) (mapcar (function +) '(1 2 3) '(4 5 6))

(b) (mapcar (function append) '((1)(2)(3))'((a)(b)(c)))

(c) (mapcar (function mapcar)
(list (function 1+) (function 1-))
'((1 2 3 4)(5 6 7 8)))

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

(defun xyzzy (x) (eval `(lambda (x) (funcall ,x (funcall ,x x)))))

(a) With what type of argument should xyzzy be call? (5)

(b) Define in a sentence what xyzzy does? (5)

(c) What will (funcall (xyzzy (function (lambda (x) (* x 10)))) 5) return? (5)

Have a good holiday break.