CS316

1. Indicate if each of the following statements are true or false.
    a. *Structured operational semantics* describes meaning of program statements by translation into intermediate-language-like code.
    b. Contiguous array memory allocation in column-major order requires more total memory space than in row-major order.
    c. Contiguous array memory allocation in row-major order gives faster array element access than column-major order.
    d. Compaction of heap memory is the process of finding variables that are undefined, i.e., have no assigned values.
    e. Garbage collection is the process of finding heap memory cells that are no longer reachable from the roots and reclaiming them in the pool of free memory space.
    f. Semantics of program statements describes their grammatical structures
    g. The mark-and-sweep garbage collection uses a graph traversal algorithm.
    h. The mark-and-sweep garbage collection never relocates reachable heap cells.
    i. The stop-and-copy garbage collection does garbage collection and heap compaction at once.
    j. In the stop-and-copy garbage collection (without use of hard disk), only half of the entire heap can be used at any time.
    k. Activation records contain memory cells for local variables of functions.
    l. Activation records are used to control function calls
    m. Dynamic links in activation records point to function return values.
    n. In object-oriented languages, like this or self field of the AR for a function call contains a pointer to the target object for the call
    o. The task of a garbage collector is to reclaim the memory space used for ARs that have been popped from the runtime stack.
    p. The scope of a variable name is the region of a program text in which the name is visible and can be used
    q. The value of a variable is the contents of the memory cell allocated to the variable.
    r. The value of a variable is the contents of the memory cell allocated to the variable.
    s. Heap-dynamic variables are allocated inside the runtime stack
    t. In statically typed languages, variables can be assigned values of any types
    u. An advantage of statically typed languages is that all or almost all of type errors can be detected at compile time
    v. Binding refers to the number of ARs in the runtime stack
    w. A function table for a class contains pointers to the codes of the functions associated with the class
    x. LISP doesn't use heap memory at all
2. Give operational semantics by intermediate code for each of the following control structures:
    a. 2-branch conditional statement: if(B) S_1 else S_2
    b. While loop statement: while(B) S
    c. Switch statement: switch(E) {case L_1 : S_1 … case L_n : S_n default : S_4 }
       Give the fall-through semantics only
    d. if (B_1) { while (B_2) S_1 } else S_2

3. Give structured operational semantics of each of the following control structures
    a. 2-branch conditional statement: if(B) S_1 else S_2
    b. While loop statement: while (B) S
    Give 3 program entities that may be bound to var names
4.
5.
    a. Give 1 advantage of statically typed languages as compared with typeless languages
    b. Give 1 advantage of typeless languages as compared with statically typed languages
    c. Of the programming languages mentioned/studied in the course, give 2 statically typed languages
    d. Of the programming languages mentioned/studied in the course, give 2 typeless languages
6. Consider the 3 categories of variables with respect to memory binding:
        i. Static
        ii. Stack-dynamic
        iii. Heap-dynamic
        For each of the following a through k, choose all categories that correctly apply to it
    a. Used in C++
    b. Used in Java
    c. Default memory binding for local variables declared in functions
    d. Default memory binding for formal parameters of functions
    e. Heap memory cells which are referenced by pointers or object-references
    f. Variables that remain bound to the same memory cells throughout the entire program execution
    g. Variables whose memory cells are subject to garbage collection
    h. Variables whose memory cells reside in the runtime stack
    i. Variables that are bound to memory cells by explicit calls to memory allocation operators (ie – constructors).
    j. Variables that are bound to memory cells before program execution
    k. Nameless memory cells that can only be accessed by chains of one or more references from the same area, the runtime stack, or the registers

7. Consider the following Java-like class definition:

```
class Node
{
    Node n;
    void f(int x)
    {
            Node a = new Node();
            a.n = this;
            if(x<=2) a.f(x+1);
    }
    void main()
    {
            Node m = new Node();
            m.f(1);
    }
}
```

   a. Give all stack-dynamic variables used in this program
   b. Would the above program create any heap-dynamic variables? If so, identify the statement(s) that would create them
   c. Trace the function calls beginning with a call to main() and give a picture that illustrates the runtime stack and its ARs at the time the stack has reached its peak size. In each AR for f, include this pointer, the parameter, and the local variable, omitting all else. Also display all of the object cells and reference pointers that have been created.

8. Consider the contiguous memory allocation for a 1-dimensional array [5 … 15].
Let BaseAddres = address([5])=10, ElementSize = 2, Give the formula for address([i_1])

9. Consider the contiguous memory allocation for a 2-dimensional array [0…10, 5…20]
Let BaseAddress = address([0,5])=10, ElementSize=1
Compute address([5,15]) in
   a. Row major order
   b. Column major order

10. Consider the row-major non-contiguous memory allocation method discussed in class
   a. Give address calculation formulas for the element [5,5] of an array [0…9,0…9].
   b. Give a picture illustrating the 1-dimensional array slices and the references involved in accessing the element [5,5,5] of a 3-dimensional array [0…9,0…9,0…9]

11. Consider the following Java-like class definitions along with main() function:

```
class A {int x; int y; void f() {…} void f2() {…}}
class B extends A {int z; void f3(){…} void f1(){…f1 redefined…}}
class C extends B {int w; void f4(){…} void f2(){…f2 redefined…}}
class D extends C {int t; void f5(){…} void f1(){…f1 redefined…}}
main() {A a=new A(); B b=new B(); C c = new C(); D d=new D();}
```

Assuming dynamic binding of function code, draw a picture illustrating the memory cells created for the variables a,b,c,d with their fields, the function tables, and all the relevant reference pointers

12. A definition of garbage memory cells have been given in terms of directed graphs of allocated memory cells
    a. What are the roots of the directed graphs and where do they reside?
    b. What are the nodes and links of the directed graphs?
    c. Define reachable nodes
    d. Define garbage nodes
13. Evaluate each of the following Scheme expressions. Just give the result
    a. (cdr (cons ‘a ‘b))
    b. (cdr (cons 1 (cons 2 ())))
    c. (car ‘(a b c d))
    d. (car (cdr ‘(a b c d)))
    e. (cons 1 (cons 2 (cons 3 ())))
14. Define the following Scheme functions
    a. (append x y) Appends list y to list x. E.g. (append ‘(1 2 3) ‘(a b c)) will return (1 2 3 a b c).
    b. (map f list) Returns the list obtained by applying the function f to each element of list.
       Example: (map square ‘(1 2 3)) will return (1 4 9).
15. Give a PROLOG program for the relation length(List, X) which holds if X is the number of elements in List
16. The following is a PROLOG program that defines the relation exist(X,List) which holds if element X exists in List
    ```
    exist(X,[X|T]).
    exist(X,[Y|T]):-exist(X,T).
    ```
    Give a complete evaluation tree for exist(3,[1,2,3]). In each step, give the unifier used.