CS 405/505 FINAL EXAM SAMPLE QUESTIONS (SOLUTIONS)

1. Fill in the Blank. A(n) deadlock is a situation where two tasks A and B each need two shared resources X and Y in order to complete their work, but A holds resource X and B holds resource Y, and each task must wait for the other to release the resource in order to continue.

2. True or False. A virtual method table is needed only for the execution of polymorphic functions in C++, but is not needed in Java.

   False. A virtual method table is also needed for Java since all Java methods are potentially polymorphic.

3. List and briefly describe two (2) conditions under which a running task may block in Ada.

   A task may initiate an entry call and there is no available accept clause to rendezvous with it.

   A task may rendezvous with an accept clause which requires some processing. In this case, the calling task blocks until the accept clause has completed.

4. Consider the Pascal array declaration:

   A : array [−10 .. 10, −5 .. 5, 1 .. 10] of T;

   Assuming that T requires 10 bytes of storage and A is stored at relative location 100, what is the relative location of A [1, 1, 2]?

   addr (A [1, 1, 2]) = addr (A) + (((1 - (-10)) * 11) + (1 - (-5))) * 10 + 2 - 1) * 10 = 100 + ((11 * 11 + 6) * 10 + 1) * 10 = 12,810

5. Define the terms *dangling reference*, *garbage*, and *memory leakage*. Describe how each is created in a program and a possible solution to each.

   See textbook.

6. Consider the following Pascal program:

```
program MAIN;
    var U, V : INTEGER;
    procedure A;
        var Y : INTEGER;
        ... (* point 4 *)
    end;
    procedure B;
        var U, V, Y : INTEGER;
        procedure C;
            ...
            begin
                ... (* point 3 *)
                A;
                ...
            end;
```

```
        begin (* B *)
            ... (* point 2 *)
            C;
            ...
        end;
    begin (* MAIN *)
        ... (* point 1 *)
        B;
        ...
    end.
```

Indicate which activation records are stacked at each of points 1, 2, 3 and 4 in the program as well as the exact contents of those activation records. Assuming static scoping and deep binding, show the contents of the display at each of these points. Assuming dynamic scoping, what is the scope of V at point 4?

Static scoping and deep binding
1) ARS = MAIN; DS = MAIN
2) ARS = MAIN, B; DS = MAIN, B
3) ARS = MAIN, B, C; DS = MAIN, B, C
4) ARS = MAIN, B, C, A; DS = MAIN, A

With dynamic scoping, V belongs to B at point 4.

7. Consider the functions below in C++-like pseudocode.

```
int I;

int F (int X, int Y) {
    I = 2;
    Y = 1;
    return X + Y;
}

void main (void) {
    int A [3];
    A [0] = 7; A [1] = 11; A [2] = 13;
    I = 0;
    cout << F (A [I], I);
}
```

Show the values stored in the **locations** named `A [0]`, `A [1]`, `A [2]`, `I`, `X` and `Y` at the entry and exit points of function `F` assuming (a) call by value, (b) call by reference, (c) call by value-result, and (d) call by name? Also, show the result which gets printed in each case.

The following table shows the results for each calling method.

Call by	A[0] i	A[0] o	A[1] i	A[1] o	A[2] i	A[2] o	I i	I o	X i	X o	Y i	Y o	Res
value	7	7	11	11	13	13	0	2	7	7	0	1	8
reference	7	7	11	11	13	13	0	1	a1	a1	a2	a2	8
value-result	7	7	11	11	13	13	0	1	7	7	0	1	8
name	7	7	11	11	13	13	0	1	t1	t1	t2	t2	12

a1 - addr (`A [0]`)
a2 - addr (`I`)
t1 - thunk (`A [I]`)
t2 - thunk (`I`)

Assuming all parameters are passed by reference, does *aliasing* occur in this program? If so, where?

`Aliasing does occur in F as I may be referred to by its own name or by formal parameter Y.`