Skill/Knowledge Check-List for CSC-1350: Intro. Computer Science - I

(Students should be able to answer the sample questions below on Programming and on JAVA)

1. (a) Name the four different parts of a variable declaration like "public int count = 0;".
(b) Do the same for different parts of a function declaration like "public boolean hasPositive(int[]nums) {···}", where we do not show the function-body (indicated by "···").
(c) Give the data-types of the items 103, 3.28, true, 'A', and "abc".
(d) State the "fundamental theorem" of programming.
(e) Explain the meaning of "define first before using" (for a variable, say) by giving an example. Does it apply for a class, how about for a function – explain your answer.

2. (a) The statement "`int as[], bs[];`" declares two arrays in one statement. Give 3 other ways to achieve the same result using one statement.
(b) Which one is the best in some sense (and why)?
(c) Is any thing wrong with the following declaration "`int[] as = new int[0];`"? What would be the value of `as`?
(d) Show the output of the following code-segment:

```
double x = 35.19152;
System.out.print("Printing x with different number of digits ");
System.out.println("after the decimal-point:");
System.out.println("%f, %2.3f, %2.2f, %2.1f\n", x, x, x, x);
```

(e) Show the results of (int)x, Math.floor(x), Math.ceil(x), and Math.round(x) for each value of *x* given below.

x	3.6	–3.6	3.5	–3.5	3.1	–3.1
(int)x						
Math.floor(x)						
Math.ceil(x)						
Math.round(x)						

Show the output of the following:

```
int[] a = {1, 2, 3, 4, 5, 6};
System.out.println("a=" + Arrays.toString(a));
```

Show the output of the following; clearly indicate the blank spaces.

```
int nums[] = { -100, -2, 1023, 1 },
    length = nums.length;
for (i=0; i<length; i++)
{   System.out.printf("%5d | %+5d | %05d ", nums[i], nums[i], nums[i]);
    System.out.printf("| %-5d | %+05d | %+-5d\n", nums[i], nums[i], nums[i]);
}
```

3. There are many parts to this question.
   (a) Identify the parts initialization, test, and update of the loop-control below; also, identify the loop-body.

```
for (sum=i=0; i<6; i++)
    if (1 == i%2) sum += i;
```

   (b) Show the value of sum after each iteration of loop in (a).
   (c) Fill-in the three missing parts "···" below so that it will compute the same final value of sum as in (a).

```
sum = ···;
for (i=···; i<6; i·····)
    sum += i;
```

   (d) Show an alternate for-loop, without loss of clarity, which will compute the same final value of sum as the one below. (Here, f(i) is a given function with integer inputs; it returns integer values.)

```
sum = 0;
for (i=0; i<n; i++)
    sum += f(i);
```

(e) Fill-in the missing parts "···" below which will give the same final value of sum, where f(i) is as in (d).

```
sum = 0;
for (i=0; i<n; i+=2)
    sum += ···;
...
```

4. There are many parts to this question.

(a) First, indicate the problem with scope of variables below and give a simple (with least modifications) corrected form. What will be the return-value for `nums[] = [-1, -2, 0, 2, 1]` now? Then, show the code after it is thoroughly cleaned (eliminating any of "break" and "continue", if possible); avoid otherwise unnecessary changes and do not increase the computations in the process.

```
boolean hasPositive(int[] nums)
{ int length = nums.length;
  for (int i=0; i<length; i++)
      if (0 < nums[i]) break;
      else continue;
  if (i < length) return(true);
  else return(false);
}
```

(b) First, show the values of remainder, small, and large after each iteration of the code below for small = 30 and large = 42; also, show the value returned by the function. Then, rewrite the for-loop using a do-while-loop and avoiding any unnecessary tests. Also, explain why for-loop is not appropriate here.

```
int gcd(int small, int large) //large >= small > 0
{ for (int i=0; i>=0; i++)
  {   int remainder = large%small;
      if (0 < remainder)
         { large = small; small = remainder; }
      else return(small);
  }
}
```

(c) Give a pair (m, n), m = small and n = large, such that there will be exactly one iteration of the original loop in (b), and also give the general condition that should hold between m and n for this to be true.

(d) Show how to construct a pair (m', n') which will have $k+1$-iterations of the loop in (b) if (m, n) causes $k \geq 2$ iterations of the loop.

(e) Show the flowcharts of the code given in (b) and of the final simplified code you obtained. How do these flowcharts justify the claim that the simplified code is superior to the original code?

5. Consider a function int[] positiveItems(int[] nums), that returns the array of positive items in `nums[]`. Thus, for `nums[] = [ 2, -1, -2, 3, -3]`, the returned-array is `[2, 3]`, which is shorter in length than `nums[]`. First, give the pseudocode for the two key steps in this function, and then give the complete code for the function.

6. Mark each method in the code below as 'R' (overriding), 'L' (overloading), or 'RL' (both) as appropriate.

```
public class A
{ public int i, j;
  public void add(int k) { i += k; j += k; }
  public int add() { return(i + j); }
  public A(int k) { i = j = k; }
  public A(int i, int j) { this.i = i; this.j = j; }
}
```

7. (a) Show the value returned by "`(new Linear(3.0, 5.0)).valueAt(1.1)` given the following.

```
public class Linear
{ private final double c1, c0;
  public Linear(double c1, double c0)
  { this.c1 = c1; this.c0 = c0; }
  public double valueAt(double x)
  { return(c1*x + c0); }
}
```

(b) Assume that you are given the class `Linear` and you are told not to modify the class, but to create two functions `double findC0(Linear linear)` and `double findC1(Linear linear)` in your program for personal use. Fill-in the missing function-bodies below; you will need only one or two statements in each function.

```
double findc0(Linear linear)
{ ...   ...
}

double findc1(Linear linear)
{ ...   ...
}
```

(c) Give the code of a function that returns the value of x such that 0 = `linear.valueAt(x)`; assume that either `c1` $\neq 0$ or `c0` = 0.

8. First, for each of the three possible outputs below, give an example x and y that will cause that output; also, give the number of evaluations of conditions of the form ">=" for that x and y.

```
if ((x >= 0) && (y >= 0))
   System.out.println("1st type");
else if (x >= 0)
        System.out.println("2nd type");
     else System.out.println("3rd type");
```

Then, rewrite the nested if-else statement that will (maximally) reduce the number of ">="-condition evaluations.

9. Write the appropriate word "null" or "default" or "parametrized" next to each constructor below.

```
public class A
{ public int first, second;
  public A() { }
  public A(int i)
  { first = second = i; }

}
```

```
public class B
{ public int first, second;
  public B()
  { first = 0; second = 1; }
  public B(int first, int second)
  { this.first = first;
    this.second = second;
  }
}
```

(a) What is the reason for giving them such names?

(b) Show the output from each of the following (using "_" to indicate spaces):

System.out.println("second = " + (new A()).second + ", second = " + (new A(1)).second);

System.out.println("second = " + (new B()).second + ", second = " + (new B(2, 3)).second);

(c) Mark each declaration of the variable `first` as 1, 2, etc. and mark each use of the variable `first` also as 1, 2, etc to indicate the particular declaration of `first` being applied.

(d) If we replace "public int first, second;" in class A by "private int first, second;" then the objects of class A becomes immutable (cannot change any of its fields/attributes) once created. In particular, we cannot create an instance of A with arbitrary values of first and second, where first $\neq$ second. What can we do, keeping the objects (instances) of type A immutable, and yet have arbitrary values for first and second?

(e) If we keep first and second "public" as in the original code for class A, and yet we want to make the instances of A immutable, what do we need to do?

(f) Is there any advantage or disadvantage of the solution in (e) instead of making first and second private?

(g) What do we need to do to achieve the same advantage or disadvantage in (f) but keeping first and second private?

(h) State the distinct properties of the following modified form of class A in comparison to the original class A?

```
public class A
{ private int first, second;
  public A() {}
  public A(int i)
  { first = second = i; }
  public A(int i, int j)
  { first = i; second = j; }
  public int getFirst()
  { return(first); }
  public int getSecond()
  { return(second); }
}
```

10. Shown below is a simple example of *permuted-index* for the input string *x* = "two very big cats chasing two big dogs" (which is an English phrase having only lower-case letters, for simplicity).

```
                          two very BIG cats chasing two big dogs
    two very big cats chasing two BIG dogs
                      two very big CATS chasing two big dogs
                 two very big cats CHASING two big dogs
two very big cats chasing two big DOGS
                                  TWO very big cats chasing two big dogs
        two very big cats chasing TWO big dogs
                               two VERY big cats chasing TWO big dogs
```

The permuted-index here consists of one output line for each word in *x*, including multiple occurrences of a word; the particular occurrence of a word related to an output line is shown in capitals. The output lines are aligned so that the capitalized words start in the same vertical position. Finally, the output lines are sorted based on the capitalized words, and the ties are broken based on the length of the part to the left of the capitalized word.
Give a clear pseudocode (with 4 to 5 steps) for creating the permuted-index from an arbitrary given phrase.

11. First, give the two key steps in the form of a pseudocode for selection sort. Then, give a refinement of the second step.

12. Show the outputs of following code-segment.

```
01 String x = "abc", y = "abc";
02 System.out.println("After initializations: x = \"abc\" and y = \"abc\"");
03 if (x == y) System.out.println("x == y test succeeds");
04 if (x == "abc") System.out.println("x == \"abc\" test succeeds");
05 if (("ab" + "c") == "abc") System.out.println("\"ab\" + \"c\" = \"abc\"");
06 x = new String("abc"); y = new String("abc");
07 System.out.println("After x = new String(\"abc\"), y = new String(\"abc\")");
08 if (x == y) System.out.println("x == y test succeeds");
09 else System.out.println("x == y test fails");
10 if (x == "abc") System.out.println("x = \"abc\" test succeeds");
11 else System.out.println("x = \"abc\" test fails");
12 if (x.equals("abc")) System.out.println("x.equals(\"abc\") test succeeds");
13 else System.out.println("x.equals(\"abc\") test fails");
```