CSC 360 Midterm 1 answer key

1. Write a recursive function which is passed two int parameters, x and y, and determines if x is a power of y (as in $y^n = x$ for some $n > 0$). Assume x and y will be greater than 1. This method should return a boolean. (8 points)

```
public boolean pow(int x, int y) {
   if(x==1) return true;          // by dividing x by y continually, we reduce it to 1
   else                           // so that we know x is only comprised of y's
         if(x%y==0) pow(x/y, y);       // otherwise if we can divide x by y, do so
                                       // and recursively call pow with reduced x
         else return false;       // dividing x by y results in a remainder so x is not
}                                 // a power of y
```

2. Given the following code from a main method and the method change, answer the questions below. Assume the `accelerate` method of Car changes some instance data(um) in Car. (9 points each)

```
int x=0;
Car c=new Car("Toyota", 2005, 10000);
change(x, c);
…

public void change(int a, Car r) {
      a++;
      r.accelerate(10);
}
```

   a. Once `change` terminates, has x changed? No, x is passed by copy so a is a different memory location.

   b. Once `change` terminates, has c changed? Yes, r and c point to the same Car

   c. We add this instruction as the first instruction to change: `r=new Car("Toyota", 2010, 0);` Once `change` terminates, has c changed? No, r now points to a different Car than c, so changing r has no impact on c.

3. We want to compute the value of `x/y * a/b` where `x`, `y`, `a` and `b` are all `int` values and both `x/y` and `a/b` have the potential of storing very large fractions so we will use the `BigDecimal` class to accomplish this. Provide the code assuming that a, b, x, y are int variables already declared and initialized and where b and y are not 0. (8 points)

```
BigDecimal d1=new BigDecimal(""+ x);
BigDecimal d2=new BigDecimal(""+ y);
BigDecimal d3=new BigDecimal(""+ a);
BigDecimal d4=new BigDecimal(""+ b);
BigDecimal d5=d1.divide(d2,10,BigDecimal.ROUND_UP);
BigDecimal d6=d3.divide(d4,10,BigDecimal.ROUND_UP);
BigDecimal d7=d5.multiply(d6);
```

4. We want to create a list of integer numbers but are not sure how long the list should be. Here are some choices.

```
      i.   int[ ] a=new int[100];
      ii.  Integer[ ] b=new Integer[100];
      iii. ArrayList<Integer> c=new ArrayList<>( );
      iv.  ArrayList<int> d=new ArrayList<>( );
```

   a. Compare these four possibilities in terms of strengths and weaknesses. (8 points)

      i. since we don't know how many int values are to be stored, this has the weakness of possibly needing array doubling or we have to limit the number of elements to 100, otherwise it is a good choice in that it is efficiently stored and utilized.

      ii. there is no value to storing the values as Integers. i is better, so we would not use ii.

      iii. in order to get around the weakness of i, we could use this approach. The weaknesses are that we do not know how ArrayList is implemented exactly so we have to use its interface, but that is not really a problem. Also, we have to use Integers to wrap our int values.

      iv. not possible because ArrayList types must be Objects.

   b. Which would you select and why? (3 points)

      Personally, I would use i and perform array doubling or limit the number of elements using if statements but I am old school. iii is a reasonable choice too and probably the better choice in terms of the "politeness" of your code in that you do not have to restrict the array's size.

5. Write a recursive method which is passed a String and a character and counts and returns the number of occurrences of the character in the String. (8 points)

```
public String count(String s, char c) {
      if(s.equals("")) return 0;
      else if(c==s.charAt(0)) return 1+count(s.substring(1), c);
      else return count(s.substring(1), c);
}
```

6. In a method we have the following. Use this to answer the two questions below (2 points each).

```
Person p;                // line 1
p = new Person(…);       // line 2
```

   a. The variable p declared in line 1 is stored in what part (area) of memory?
      The reference variable itself is stored on the run time stack.

   b. The object instantiated in line 2 is stored in what part (area) of memory?
      The object being referenced is stored in the heap.

7. We want to use the WritingUtensil class below to extend it to `BallPointPen` which adds an instance datum, `boolean hasCap`.

```
public class WritingUtensil {
      private boolean erasable, refillable; // instance data
      private String type, writingMaterial; // instance data

      public WritingUtensil(...) {…} // this constructor receives
                             // values for all 4 instance data
                             //  and sets them as you would expect
      public String toString( ) { return type + "\t" +
         writingMaterial + "\t" + erasable + "\t" + refillable;}
      // accessors and mutators available for each instance datum
}     //       as you would expect
```

a. Describe any changes you should make to `WritingUtensil` before extending it. (6 points)

First, the instance data have to be protected, not private.

Second, we would want to implement a no-arg constructor.

b. Define the BallPointPen subclass overriding and adding methods as necessary. (10 points)

```
public class BallPointPen extends WritingUtensil {
      protected boolean hasCap;

      public BallPointPen( ) {
            super( );
            hasCap=false;    // assume false if we do not know
      }

      public BallPointPen(boolean e, boolean r, String t,
                  String w, boolean h) {
            super(e, r, t, w);
            hasCap=h;
      }

      @Override
      public String toString( ) {
            return super.toString( ) + "\tHas cap:" + hasCap;
      }

      public boolean getHasCap( ) { return hasCap;}
      public void setHasCap(boolean h) {hasCap=h;}
}
```

8. What does `final` mean when applied in these contexts? (3 points each)

a. to a method – the method cannot be overridden in a child class

b. to a class – the class cannot be extended into a child class

9. Recall from program 1 that we had Fighter, Weapon and Armor classes. We want to extend each of these to have MagicalFighter, MagicalWeapon and MagicalArmor classes. For MagicalFighter, we add an instance datum `boolean magicDampener` and for the MagicalWeapon and MagicalArmor classes we add int instance data of `magicalDamage` and `magicalProtection`. The idea is that if a Fighter has a magical weapon or armor then it adds to the possible damage or protection, unless the Fighter they are fighting has the ability to dampen magic. Only a MagicalFighter has the possibility of dampening magic but not all MagicalFighters will be able to do so. Answer the following questions. (12 points)

   a. What methods from each of the Fighter, Weapon and Armor classes will you have to override in order to implement these subclasses appropriately?

      For all 3, any toString methods since we have additional instance data
      For Fighter, the fight method because we have to test to see if there is magic involved
      For Weapon, getMaxDamage and isBetter
      For Armor, getProtection and isBetter

   b. Aside from the constructor, what other method(s) should be implemented in MagicalFighter, MagicalWeapon and MagicalArmor?

      We now need accessors for the new instance data to return a boolean as to whether the item is magical (or the Fighter has the ability to dampen magic)

   c. If done correctly, would you also have to modify your main method in your AdventureGame class? Briefly explain.

      If any of the Fighters, Weapons or Armor use magic, we would have to modify the instantiation but that would be the only thing to modify. We would not for instance have to change any method calls and we would not have to modify the declaration of any variables because of polymorphism.

10. The following is a partial solution to the Water Jugs problem (most of the recursive calls have been omitted to save space). The variable `list` is an `ArrayList<String>` declared as an instance datum of the class. What is `list` used for and what would happen if we did not have it (and thus removed the first if-else statement)? (6 points)

```
public boolean jugs(int x, int y)  {
        boolean temp;
        if(list.contains(""+x+y)) return false;
           else list.add(""+x+y);
        if(x<0||x>4||y<0||y>3) return false;
        else if(x==2) return true;
        else {
            temp=jugs(x,0);
            if(!temp) temp=jugs(0,y);
            if(!temp) temp=jugs(4,y);
            ...   // other water jug moves here
        }
        return temp;
    }
```

List is used to record every state visited to prevent us from revisiting the same state twice. States are recorded as a String storing “xy” as in “40” or “03”.

Without list, we have the potential of revisiting the same state repeatedly without detecting it. This would lead to infinite recursion which would eventually result in a StackOverflowException.

11. Given the following UML, provide the code to implement this entire class. The computeDEF method is called from both constructors and from setABC. The computeDEF method sets def by assigning it the value of abc * 2 if abc is positive, abc if abc is negative, or 1 if abc is 0. toString should return both instance data. (12 points)

```
Foo
-abc: int
-def: double
---------------------
+Foo( )
+Foo(abc: int, def: double)
+getABC( ): int
+setABC(abc: int);
-computeDEF( ): void
+toString( ): String;
```

```
public class Foo {
      private int abc;
      private double def;

      public Foo( ) {
            abc=0;
            computeDEF( );
      }

      public Foo(int a) {
            abc=a;
            computeDEF( );
      }

      public int getABC( ) { return abc; }

      public void setABC(int a) {
            abc=a;
            computeDEF( );
      }

      private void computeDEF( ) {
            if(abc>0) def=2.0*abc;
            else if(abc==0) def=1.0;
            else def=abc;
      }

      public String toString( ) {
            return “” + abc + “\t” + def;
      }
}
```