Exam 2 Practice Problems

Main topics: Recursion, class design with inheritance

Note: these questions are not intended to reflect the style, length, or difficulty of the 2nd midterm overall *absolutely*. However, you may assume that I will not ask anything too much harder than what you see here. Additionally, any harder questions on the midterm will probably be based on problems that appear here.

1. The following formula defines what are called binomial coefficients. These numbers arise in various mathematical situations:

   ```
         B(n, k) = 1                    if n=k or k=0
   and   B(n, k) = (n/k) * B(n-1, k-1)  otherwise
   ```

   Note that the formula only works for nonnegative values of n and k, where n > k.

   (a) Translate this function formula into a recursive Java function.

   (b) In your Java function, identify the base case and the recursive case.

   (c) Draw a recursive diagram illustrating how B(5, 3) would be calculated. You should show which recursive calls are generated by which other recursive calls, and the return value of each call.

   (d) How many total recursive calls are there to calculate B(5, 3), including the first one?

2. The following formula defines a made-up recursive function:

   ```
         f(a, b) = a + b              if a <= b
   and   f(a, b) = (a-b) + f(a-2, b-1)  otherwise
   ```

   (a) Translate this function formula into a recursive Java function.

   (b) In your Java function, identify the base case and the recursive case.

   (c) Draw a recursive diagram illustrating how f(6, 3) would be calculated. You should show which recursive calls are generated by which other recursive calls, and the return value of each call.

   (d) How many total recursive calls are there to calculate f(6, 3), including the first one?

   (e) Suppose we change the part of the function definition above that says f(a-2, b-1) to f(a-1, b-2). What happens when we call f(6, 3) now?

   (f) What are the only legal values I can call the "f" function on now that will not cause the problem in part (e)?

3. Recall that a palindrome is a string that reads the same forwards and backwards. Suppose you are given the following recursive function that is used to **generate** palindromes:

```
public static String makePal(int i) {
   if (i == 0) {
      return "A";
   }
   else if (i == 1) {
      return "B";
   }
   else {
      return makePal(i – 1) + makePal(i – 2) + makePal(i – 1);
   }
}
```

(a) What string is returned from the call makePal(2)?

(b) What about makePal(3)?

(c) What is the length of the string returned from makePal(5)? Hint: Do not actually compute this string.

4. Write a recursive function to count the number of uppercase letters in a string. Assume there is a boolean function called isUpper() that takes a char argument that returns true if its argument is uppercase, and false if not.

```
public static int countUpper(String str) {
   // write code here --- no loops!
}
```

5. Here is the code for binary search:

```
public static int binarySearch(ArrayList<Integer> array, int key, int low, int high)
{
  if (low > high)
    return -1; // not found
  else
  {
    int mid = (low + high) / 2; // notice integer division

    if (array[mid] == key)
      return mid;
    else if (array [mid] > key)
      return binarySearch(array, key, low, mid-1);
    else
      return binarySearch(array, key, mid+1, high);
  }
```

Suppose you are given this array:

index:	[0]	[1]	[2]	[3]	[4]	[5]	[6]
value:	1	4	8	14	28	39	41

You want to use the binary search algorithm to find whether or not the number 28 is in the array. Draw a recursion diagram illustrating the function calls that would be used. Make sure your diagram shows the values of low, high, and mid, as well as the return values from each call.

6. You are the owner of an ice cream parlor. With so many places now passing laws that restaurants have to post calorie counts for their products, you decide to write a program to help you calculate these numbers.

   (a) Create a class to represent an ice cream sundae. This class should contain:
   - One instance variable, representing the number of scoops in the sundae.
   - One public constructor, taking one integer argument that represents the number of scoops desired in this sundae.
   - One public method, taking no arguments, which returns an int representing the number of calories in the sundae. Google says one scoop of ice cream has 137 calories.

   Example of using the class:

```
Sundae yummy = new Sundae(2);          // create a 2-scoop sundae
System.out.println(yummy.getCalories());    // prints 274 calories
```

   (b) Create a class to represent a banana split. This class should inherit from your ice cream sundae class. A banana split is simply an ice cream sundae that also contains a number of sliced bananas. This class should contain:

   - One instance variable, representing the number of bananas in the sundae.
   - One public constructor, taking two integer arguments that represent the number of desired ice cream scoops and bananas, respectively, in the sundae.
   - One public method, overriding the method in Sundae, to return the number of calories in the banana split. Google says one small banana has 90 calories.
     - Hint: to implement this, you should not duplicate any code from the Sundae getCalories() method. Instead, use a technique from class to call the superclass's version of getCalories().

   Example of using the class:

```
BSplit tasty = new BSplit(3, 1); // create a 3-scoop, 1-banana sundae
System.out.println(tasty.getCalories());    // prints 501 calories
```

7. Consider the classes Boat and Submarine:

```
public class Boat {
   protected int fuel = 2;

   public void aheadFull() {
      fuel--;
   }

   public int getFuel() { return fuel; }
}

class Submarine extends Boat {
   public void aheadFull() {
      super.aheadFull();
      fuel–;
   }

   public void refuel(int x) {
      fuel += x;
   }
}
```

Assume we have a main function:

```
public static void main(String[] args) {
   Boat f = new Boat();
   Submarine g = new Submarine();


   f.aheadFull();      // 1
   g.aheadFull();      // 2

   f.refuel(10);   // 3
   g.refuel(10);   // 4


}
```

(a) Which of the lines in main() with the comments 1-4 are legal in Java and which ones will be flagged as errors (have red squiggles under them)?

(b) Assume you comment out any line that would cause an error. After these lines run, how much fuel does each Boat/Submarine have?

8. Consider the classes Restaurant and Buffet:

```
public class Restaurant{
   protected int cost = 2000;

   public void printCosts() {
      System.out.println("Cost: " + cost);
   }

   public void printMenu()
   {
     System.out.println("Menu: Chicken, fries, burgers, cake");
   }
}

class Buffet extends Restaurant {
   private int extraFoodCosts = 50;

   public void printCosts() {
      System.out.println("Cost: " + (cost + extraFoodCosts));
   }

   public void ratingsCompetition(Restaurant A) {
      System.out.println("Competition is rumbling…");
      cost = cost + 300;
       A.printCosts();
   }

   public void printMenu() {
      System.out.println("Menu: Chicken, rice, tofu, pizza, scones");
   }
}
```

Assume we have a main function:

```
public static void main(String[] args) {
   Restaurant bk = new Restaurant();
   Restaurant mcd = new Restaurant();
   Buffet rat = new Buffet();
   Buffet rs = new Buffet();

   bk.printMenu();              // 1
   bk.printCosts();             // 2
   bk.ratingsCompetition(mcd);  // 3

   rat.printMenu();             // 4
   rat.printCosts();            // 5
   rat.ratingsCompetition(mcd); // 6
   rat.printCosts();            // 7
   mcd.printCosts();            // 8

   rs.ratingsCompetition(rat);  // 9

}
```

For each line in main, record A.) if the program would generate an error and B.) what the output of the program would be.