CS 11 Fall 2000 — **Sample** mid-term

1. Consider the Java code below and answer the questions that follow.

```
System.out.print("Enter x: ");
int x = Keyboard.readInt();
System.out.print("Enter y: ");
int y = Keyboard.readInt();

if (x >= y) {
  System.out.println("Flip " + x);
} else if (y < 8) {
  while (x < y) {
    System.out.println("Quip " + x);
    x = x + 1;
  }
}
if (x == y) {
  System.out.println("Blip " + y);
}
```

   (a) What would be the output if the user entered `8` and `8`?

   (b) What would be the output if the user entered `3` and `5`?

2. Now consider a different code fragment:

```
char[] x = {'z', 'i', 'n', 'g'};
char[] y = {'p', 'i', 'n', 'g'};

y[0] = 'z';

if (compareArray(x, y)) {
  System.out.println("Yes!");
}

if (x != y) {
  System.out.println("No...");
}
```

What is the output of this code fragment?

3. Provide short (at most a few sentences) answers to **any two of the three** following questions:

   (a) What does it mean for a program to be *robust?*

   (b) What does it mean for a method to be *recursive?*

   (c) Is *linear search* faster than *binary search* in every instance? Briefly support your answer.

4. Given an array of numbers, the *median* is the *middle element taken from those sorted numbers.* In other words, if an array contained {`3.6, 1.0, 2.0`}, the average would be **2.2**, but the median would be **2.0**, as that is the middle element when those numbers are sorted. **Write a method** that accepts a pointer to an array of `double`, and returns the median. Note that the array being passed to this method is not necessarily sorted. (If there an even number of elements, your method may return either one of the elements that are in the middle.)

5. The *Fibonacci sequence* is a recursive sequence of numbers, where the n^{th} number in the sequence is the sum of the two previous numbers in the sequence. So:

```
F(0) = 1 // The 0-th Fibonacci number is 1
F(1) = 1 // The 1-st Fibonacci number is 1
F(n) = F(n-1) + F(n-2) // The n-th Fibonacci number
                       // is the sum of the
                       // previous two numbers.
```

So, the first few elements of the sequence look like this:

```
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...
```

Write a recursive method named `fib` that calculates and returns the n^{th} Fibonacci number, where n is accepted as a parameter.