1 Our First Java Program

Below is our first Java program of the semester. Next to each line, write out what you think the code will do when run. If you think the line will result in an error, correct it, and proceed through the code as if it is running your corrected version. *This exercise is adapted from Head First Java.*

```
1   size = 27;        int
2   name = "Fido";    String
3   Dog myDog = new Dog(name, size);
4   Dog yourDog = new Dog("Scruffy", 1000);
5   Dog[] dogList = new Dog[3];
6   dogList[0] = myDog;
7   dogList[1] = yourDog;
8   doglist[2] = 5;      X  = new Dog ( name , 5)
9   dogList[3] = new Dog("Cutie", 8)  X  走位置
10  int x;
11  x = size - 5;
12  if (x < 15) {
13      myDog.bark(8);
14  }
```

2   Mystery

This is a function (a.k.a. method). It takes an array of integers and an integer as arguments, and returns an integer.

```java
1   public static int mystery(int[] inputArray, int k) {
2       int x = inputArray[k];
3       int answer = k;
4       int index = k + 1;
5       while (index < inputArray.length) {
6           if (inputArray[index] < x) {
7               x = inputArray[index];
8               answer = index;
9           }
10          index = index + 1;
11      }
12      return answer;
13  }
```

(a) Describe what `mystery` returns if `inputArray = [3, 0, 4, 6, 3]` and `k = 2`.

(b) Can you explain in plain English what `mystery` does?

找到从k位置 到末尾的 最小值的 下标

Extra: This is another function. It takes an array of integers and returns nothing.

```java
1   public static void mystery2(int[] inputArray) {
2       int index = 0;
3       while (index < inputArray.length) {
4           int targetIndex = mystery(inputArray, index);
5           int temp = inputArray[targetIndex];
6           inputArray[targetIndex] = inputArray[index];
7           inputArray[index] = temp;
8           index = index + 1;
9       }
10  }
```

Describe what `mystery2` does if `inputArray = [3, 0, 4, 6, 3]`.

0 3 4 6 3

0 3 3 6 4

0 3 3 4 6

3   Writing Your First Program

Implement `fib` which takes in an integer n and returns the *n*th Fibonacci number. You may not need to use all the lines.

The Fibonacci sequence is $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$. The first two numbers in the sequence are 0 and 1, and every number thereafter it is the sum of the two numbers in the sequence before it.

```
public static int fib(int n) {
```

```
        if(n== 1)   return 0;
        if(n==2)  return 1;
        n-=2
        int a = 0, b=1, res=0
        while ( n>0) {
            res = a+b
            a = b
            b = res
            n--
        }
        return res
}
```

$$\text{If } (n \leq 1)$$
$$\quad \text{return } n$$
$$\text{else}$$
$$\quad \text{return } fib(n-1) + fib(n-2)$$

Extra: Implement a more efficient version of `fib` in 5 lines or fewer. Here, efficiency might mean making less recursive calls or doing less overall computation. You don't have to make use of the parameter k in your solution.

```
public static int fib2(int n, int k, int f0, int f1) {
```

```
        if(n == 1)  return f0
        if( n==2)  return f1.
        return fib2( n, f0,f1) + fib2 (n-2, f0,f1)
```

$$\text{If } (n == k)$$
$$\quad \text{return } f0$$
$$\text{else}$$
$$\quad \text{return } fib2(n, k+1, f1,$$
$$\qquad\qquad\qquad\qquad f0+f1)$$

```
}
```