Name: ______________________

Directions: Work only on this sheet (on both sides, if needed); do not turn in any supplementary sheets of paper. There is actually plenty of room for your answers, as long as you organize yourself BEFORE starting writing.

1. (30) Following is a table of analogies between the C/Intel/Linux world and the JVM world. Fill in the blanks! (The lengths of the blanks are not meaningful.)

<table>
<thead>
<tr>
<th>C/Intel/Linux</th>
<th>JVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer.parseInt()</td>
<td>invokevirtual, invokevirtual</td>
</tr>
<tr>
<td>pushl $1</td>
<td></td>
</tr>
<tr>
<td>.o or ELF</td>
<td></td>
</tr>
<tr>
<td>EBP</td>
<td></td>
</tr>
<tr>
<td>ESP</td>
<td></td>
</tr>
</tbody>
</table>

2. (20) Consider the function gy() on p.235. Fill in the blanks: The reason that x is in slot 1 instead of slot ___________ is that the Java keyword ___________ is not present in line 1.

3. (50) Consider this Java code to find Fibonacci numbers. (These are 1,1,2,3,5,8,13,21,..., each one being the sum of the previous two.)

```java
public class Fib {
    public static void main(String[] clargs) {
        int i, n, fib[];
        n = Integer.parseInt(clargs[0]);
        fib = new int[n];
        fib[0] = fib[1] = 1;
        genfibs(n, fib);
        for (i = 0; i < n; i++)
            System.out.println(fib[i]);
    }

    public static int genfibs(int k, int fbs[]) {
        int i;
        for (i = 2; i < k; i++)
            fbs[i] = fbs[i-1] + fbs[i-2];
        return 0;
    }
}
```

Below is part of the output from running this through javap -c. Fill in the blanks.

```java
public static void main(String[] clargs) {
    int i, n, fib[];
    n = Integer.parseInt(clargs[0]);
    fib = new int[n];
    fib[0] = fib[1] = 1;
    genfibs(n, fib);
    for (i = 0; i < n; i++)
        System.out.println(fib[i]);
}
```

Solutions:

1.  

<table>
<thead>
<tr>
<th>C/Intel/Linux</th>
<th>JVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>atoi()</td>
<td>invokevirtual, invokevirtual</td>
</tr>
<tr>
<td>pushl $1</td>
<td></td>
</tr>
<tr>
<td>.o or ELF</td>
<td></td>
</tr>
<tr>
<td>EBP</td>
<td></td>
</tr>
<tr>
<td>ESP</td>
<td></td>
</tr>
</tbody>
</table>

2. 0, static

Key points:
- fib is in slot 3, so result of newarray must go there
- for the call to genfibs(), we must push n and then (the address of) fbs onto the stack, and n is in slot 2
- to execute if_cmpge we must first push its two operands, i and k
- the compare operation is there to test whether we’ve finished the loop, and if so, we jump down to the return 0, in offsets 27 and 28 in genfibs()
- similarly, offset 24 in genfibs() is the bottom of the loop; if we’re not done, we jump to the top, offset 2
public Fib();

Code:
0:  aload_0
1:   invokespecial    #1; //Method java/lang/Object."<init>":()V
4:   return

public static void main(java.lang.String[]);

Code:
0:  aload_0
1:   iconst_0
2:   aaload
3:   invokestatic    #2; //Method java/lang/Integer.parseInt:(Ljava/lang/String;)I
6:   istore_2
7:   iload_2
8:   newarray int
10:  astore_3
11:  aload_3
12:  iconst_0
13:  aload_3
14:  iconst_1
15:  iconst_1
16:  dup_x2
17:  iastore
18:  iastore
19:  iload_2
20:  aload_3
21:  invokestatic    #3; //Method genfibs:(I[I)I
24:  pop
25:  iconst_0
26:  istore_1
27:  iload_1
28:  iload_2
29:  if_icmpge 47
32:  getstatic    #4; //Field java/lang/System.out:Ljava/io/PrintStream;
35:  aload_1
36:  iload_1
37:  iastore
38:  invokevirtual    #5; //Method java/io/PrintStream.println:(I)V
41:  iinc 1
42:  goto 27
47:  return

public static int genfibs(int, int[]);

Code:
0:   iconst_2
1:   istore_2
2:   iload_2
3:   iload_0
4:   if_icmpge 27
7:   aload_1
8:   iload_2
9:   aload_1
10:  iload_2
11:  iconst_1
12:  isub
13:  iastore
14:  iload_1
15:  iload_2
16:  iconst_2
17:  isub
18:  iastore
19:  iadd
20:  iastore
21:  iinc 2
22:  goto 2
24:  iconst_0
27:  ireturn