1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	T

LAST NAME	
FIRST NAME	
STUDENT NUMBER	

B

COMP 249 Midterm Test

Winter 2015
Section PP

Duration: 75 minutes
No calculators or additional resources

Write only on the provided sheets, however you may write on the back of each question sheet if you need extra space

[01] (1 mark) Consider the following code snippet. Specify the output or exception generated by the following code.

```
int[] a={3,1,3,3,7};
int[] b={4,1};
int[][] c={a,b};
System.out.println(b[a[1]-b[1]]);
```

b[1-1]=b[0]=4

[02] (1 mark) What will the output/exception be if the last line of Question [01] is replaced with:

```
System.out.println(c[a[1]][b[1]]);
```

c[1][1]=b[1]=1

[03] (1 mark) What will the output/exception be if the last line of Question [01] is replaced with:

```
System.out.println(c[a[2]][1]);
```

c[3][1]=Out of bounds

[04] (1 mark) What will the output/exception be if the last line of Question [01] is replaced with:

```
System.out.println(c[1][c.length-1]);
```

c[1][2-1]=c[1][1]=b[1]=1

[05] (2 marks) Consider the following three commented lines:

```
int[] a={};
int[] b;
int[] c=null;

/* 1 */ System.out.println(a.length);
/* 2 */ System.out.println(b.length);
/* 3 */ System.out.println(c.length);
```

Which line(s) result in a compilation error?

2 b/c b is uninitiated. 1 is fine and 2 will generate a runtime error (null pointer exception)

[06] (4 marks) Consider the following method:

```
public static void MethodA (int x) {
          for(int i=0;i<=x;i++){
          System.out.println(i);
}
```

Write the equivalent method recursively.

```
public static void MethodB (int x) {
     if (x>0) MethodB(x-1);
     System.out.print(x);


}
```

Assume the value of `a` is known at compile time. Assuming the compiler does not optimize the code: (a) which is faster, `MethodA` or `MethodB`? (b) describe the fastest possible implementation.

a) MethodA is faster than MethodB because it does not create new method calls
b) Uncurling the loop in MethodA is the fastest

Note: I dropped (b), and gave 1 mark for this part and 3 for the recursion. If you get (b) correct, you get a bonus mark.

[07] (3 marks) Fill in the three black squares such that `SomeMethod` returns the number of even numbers in a list of integers: *e.g.,* `SomeMethod(1,2,3,4)` returns `2` and `SomeMethod(3,1,3,3,7)` returns `0.`

```
public static int SomeMethod ( [redacted] ) {
     int l = [redacted];
     int j = 0;
     for (int i=0; i<l; i++){
          if ([redacted]) j++;
     }
     return j;
}
```

int… a
a.length
a[i]%2==0

[08] (4 Marks) The object `Vault` contains a protected int named `code`. Write a method `stealCode` to print the code in `target` to the screen assuming `Steal` is in a different package than `Vault`. You may use a `Helper` object.

```
public class Vault {
     private int code;
     public Vault () {this.code=0;}
     public Vault (int c) {this.code=c;}
     public Vault (Vault v) {this.code=v.code;}
     protected int getCode() {return this.code;}
}

public class Steal {
     public void stealCode(Vault target){
          [redacted];
     }
}

public class Helper [redacted] {
     public Helper ( [redacted] ){
          [redacted];
          System.out.println([redacted]);
     }
}
```

```
public class Steal {
     public void stealCode(Vault target){
          new Helper(target);
     }
}

public class Helper extends Vault {
     public Helper (Vault v){
          super(v); /* Most commonly missed line (-1) */
          System.out.println(v.getCode());
     }
}
```

[09] (6 Marks) An object `Child` extends the object `Parent` as follows.

```
class Parent {
     public Parent () {
          this.hello();}

     /*A*/ void hello() {
          System.out.println("Hello");}
}

class Child extends Parent {
     public Child () {
          this.hello();}

     /*B*/ void hello() {
          System.out.println("Bonjour");}
}
```

Consider the following code snippet:

```
Child c = new Child();
c.hello();
```

Options for how the black boxes labeled /*A*/ and /*B*/ might be filled in are shown below. For each column, specify what the output of the above code will be for the given values of /*A*/ and /*B*/.

/*A*/	public	public static	private
/*B*/	public	public static	public
Output	Bonjour Bonjour Bonjour	Hello Bonjour Bonjour	Hello Bonjour Bonjour

[10] (2 Marks) Refer to the same objects as Question [9]. Assume that both of the black boxes labelled /*A*/ and /*B*/ are filled in with `public static`. What is the output of the following code snippet?

```
Parent c = new Child();
c.hello();
```

Hello
Bonjour
Hello

[11] (4 Marks) Refer to the Parent and Child objects in Question [9]. You can ignore the method and just consider the constructor. For each of the following, specify if the line of code is valid or invalid (i.e., generates an error at either compile- or run-time).

Code Snippet	*Circle One*
`Child c = new Child();`	
`Parent p = new Child();`	Valid Invalid
`p instanceof Child;`	True False
`c = (Child) p;`	Valid Invalid
`c = (Child)(new Parent());`	Valid Invalid

[12] (6 Marks) Trace through the following code and specify its output

```
class NegativeException extends RuntimeException {}
class ZeroException extends NegativeException {}

class Driver {

static boolean Tower (int a) {
     try{
          if (a<0) throw new NegativeException();
          else if (a==0) throw new ZeroException();
          else if (a>=553) return true;
          else return false;
     }
     catch (ZeroException e){
          System.out.println("Use natural number");}
```

```
        finally{
             System.out.println("One last thing");
        }
        System.out.println("All done.");
        return false;
    }

    public static void main(String[] args) {
        System.out.println(Tower(0));
        System.out.println(Tower(1000));
        System.out.println(Tower(-5));

    }//main
    }//Driver
```

```
Use natural number
One last thing
All done.
false
One last thing
true
One last thing
Exception in thread "main" NegativeException
        at Driver.Tower(Driver.java:9)
        at Driver.main(Driver.java:28)
```

Note: You do not have to specify the exception, only that one is thrown

[13] (1 Mark) In Question [12], consider adding the line:

```
    catch (NegativeException e){
        System.out.println("Use positive number");}
```

Where could this line be added?

After catching ZeroException, otherwise it will catch both NegativeException and ZeroException since ZeroExceptions are NegativeExceptions.

[14] (4 Marks) Consider the following code snippet. It produces a compilation error. Specify exactly what changes need to be made to correct the error (without introducing new errors/exceptions).

```
try{
     PrintWriter f = new PrintWriter("file.txt");
     f.println("Start of file");
     System.out.println(“Written");
}

catch (FileNotFoundException e){
     System.out.println("Hmmm...");
     File ff = new File("file.txt");
     if (ff.isDirectory())
          System.out.println("Problem found");
     else if (!ff.canWrite())
          System.out.println("Problem found");
     else System.out.println("Problem not found");
}

finally{f.close();}
```

```
PrintWriter f = null;
try{f = new PrintWriter(“file.txt");
…
…
…
finally{if (f!=null) f.close();}
```

[15] (2 Marks) In Question [14], when Alice runs the corrected program, she receives the output:

```
Hmmm...
Problem found
```

Explain each scenario that could have led to her receiving this output.

```
file.txt is read-only or file.txt exists but is a directory
```

[16] (2 Marks) In Question [14], show how to change the code so that the file is appended to instead of overwritten.

```
PrintWriter f = new PrintWriter(new
FileOutputStream("file.txt",true));

A second solution is to use Random Access Files and move the
file pointer to the end of the file
```