CS 61B – Summer 2005 – (Porter)

Midterm 1 – July 6, 2005 – SOLUTIONS

Do not open until told to begin

This exam is CLOSED BOOK, but you may use 1 letter-sized page of notes that you have created.

Problem 0: (1 point) Please fill out this information, and when told to begin, please put your name on each page of the exam.

Your name:	Solutions
Your cs61b login:	
Lab time:	
Lab TA's name:	
Name of person to your left:	
Name of person to your right:	

Do not write below here:

Problem	**Score**	**Total possible**
0	1	1
1 (5 minutes)	9	9
2 (10 minutes)	15	15
3 (15 minutes)	15	15
4 (20 minutes)	30	30
5 (10 minutes)	15	15
Total:	85	**85**

Do not open until told to begin

Good Luck!

Problem 1 (9 pts) – General – (5 minutes)

A. (3 pts) What does the following code fragment print?

```
int a = 3;
int b = a + 7;
System.out.println(b);
```

10

B. (3 pts) What is the value of `x` after this code segment executes:

```
public void set(String s) {
    s = "Changed";
}
...
String x = "Original";
set(x);
```

1. "Original"
2. "Changed"
3. Undefined

1 – Original

C. (3 pts) Consider the following definitions.

int i = 3;
long k = 10L;
float f = 3.14f;
double d = 2.17;

Given those definitions, indicate whether or not each of the following produces a compile-time error.

i += 20; **Compiles** / Does not compile

f = d; Compiles / **Does not compile**

f = (float) ((double) f / d); **Compiles** / Does not compile

Problem 2 (15 pts) – Conditionals – (10 minutes)

A. (10 pts) Fill in the body of the **numVowels** method below so that it returns the number of vowels in its argument `word`. Words are represented as arrays of **char**s. The set of vowels is: **{a, e, i, o, u}**. The argument **lowerCaseWord** in will only have lower case letters.

The only conditional statement that your method may use is the 'switch' statement. Do *not* use an 'if' statement in your answer.

```
public int numVowels( char [] lowerCaseWord )
{
     int numVowels = 0;

     for (int i = 0; i < lowerCaseWord.length; i++) {
          switch (lowerCaseWord[i]) {
               case 'a':
               case 'e':
               case 'i':
               case 'o':
               case 'u':
                         numVowels++;
                         break;
               default:
                         break;
          }
     }
     return numVowels;
}
```

B. (5 pts) Consider the following code:

```
int i = 1;

do {
     while (i % 4 != 0)
          i++;
     System.out.println(i);
     i = i * 2;
} while ( i != 16 );
```

What is the output? ___________________________**4**___ **8**_____________________

Problem 3 (15 pts) – Arrays – (20 minutes)

A. (15 pts) Fill in the following method body so that it reverses an array of integers. **Your method is to return a new array—do not modify the array passed into your method. Do not make use of the Java class library (Vector, ArrayList, etc).**

You may assume that the `ar` array is of even length.

```
public int[] rev( int [] ar ) {
     int [] revar = new int[ar.length];

     for (int i = 0; i < ar.length; i++) {
          revar[i] = ar[ar.length - i - 1];
     }
     return revar;
}
```

Problem 4 (30 pts) – Defining/using classes and objects (20 mins)

Recall the Library class from lecture. A slightly modified version is attached to the end of this midterm for your reference. Its constructor has been modified so that Libraries have names. Additionally, some of ArrayList's methods have been listed as well. A library consists of books, which may be on the shelf or may be checked out. We will now consider a *Library System.* A library system is a set of Libraries. For example, the University of California Berkeley has the Doe, Bancroft, Moffitt, and over twenty subject-specific libraries.

You are going to write some of the methods for a **LibrarySystem** class. Here is a skeleton of what it will look like:

```
public class LibrarySystem {
        private ArrayList libraries;

        public LibrarySystem() {
                libraries = new ArrayList();

                libraries.add(new Library("Doe");
                libraries.add(new Library("Moffitt");
                libraries.add(new Library("Bancroft");
        }

        public boolean bookIsInSystem(String title) { ... }

        public String findBook(String title) { ... }

        public Book checkOut(Book book) { ... }

        public void checkIn(Book book, String libraryName) { ... }
}
```

A. (15 pts) Fill in the body of the **bookIsInSystem** method. It takes a String representing the book's title and searches through each library to see if the book is in the system.

```
public boolean bookIsInSystem(String title) {

     for (int i = 0; i < libraries.size(); i++) {
          Library l = (Library) libraries.get(i);
          if (l.isOnShelf(title)) {
               return true;
          }
     }
     return false;

}
```

B. (15 pts) Fill in the body of the **checkIn** method. It takes a reference to a Book object, as well as the name of a library, and it checks the book into that library. You can assume for this problem that *libraryName* refers to a valid Library that has been properly put into the LibrarySystem.

```
public void checkIn (Book book, String libraryName) {

     for (int i = 0; i < libraries.size(); i++) {

          Library l = (Library) libraries.get(i);

          if (l.toString().equals(libraryName)) {

               l.checkIn(book);
          }
     }
}
```

Problem 5 (15 pts) – Inheritance – (10 minutes)

Consider the following classes:

```
public class A {
        public int getX() {
                return 3;
        }
}

public class B extends A {
        public int getX() {
                return 5;
        }

        public boolean someFun() {
                // some function
        }
}

public class C extends B {
}
```

A. (10 pts) Consider the following code fragment:

```
B objb = new B();
A obja = objb;
```

What is returned by obja.getX()? _____________________5________

What is returned by objb.getX()? _________________5____________

B. (5 pts) Given these definitions:

```
A a = new A();
B b = new B();
C c = new C();
```

Given those definitions, indicate whether or not each of the following produces a compile-time error:

a.getX(); **Compiles** / Doesn't compile

c.getX(); **Compiles** / Doesn't compile

a.someFun(); Compiles / **Doesn't compile**