CIS 110 Summer 2012 Final Exam, 28 June 2012, Answer Key

Miscellaneous

1. (1 points)

   (a) Write your name, recitation number, and PennKey (username) on the front of the exam.

   (b) Sign the certification that you comply with the Penn Academic Integrity Code

Short Answer

2. (5 points) Answer each of the following questions in **at most** two sentences.

   (a) If I pushed 3, then 5, then 7 onto a stack of integers, what would `pop()` remove?

   7

   (b) What are the best and worst case running times for insertion sort, and in which cases do they occur? If you expect the input to be random, what do you expect the running time to be? You may use any notation you like as long as it is clear that you understand the answer.

   The best case is $O(n)$ for a sorted array and the worst case is $O(n^2)$ for an array in reverse order. If the input is random, the expected running time is $O(n^2)$.

   (c) What is one reason we would prefer a linked list to an array? What is one reason we would prefer an array to a linked list?

   Linked lists can grow or shrink at will. Array have random access and use less memory than linked lists for the same number of elements.

   (d) Give two reasons why we might want to use an interface.

   Interfaces allow the Java compiler to enforce program API specifications and also allow a single method to operate on a wide range of classes that provide a conceptually similar set of operations with very different implementations.

   (e) What is the purpose of the `Comparable` interface?

   It allows methods that rely only on the ability to compare elements, such as sorting and searching, to work on object of any class where it is meaningful to compare elements, including user-defined classes.

Debugging

3. (7 points) Find and correct 7 errors in the following code. Note: some of the bugs may cause incorrect behavior rather than syntax errors. The line numers are included for your convenience.

```
 1: public class Ouch {
 2:   private int[] arr;
 3:   private string name;

 4:   public Ouch() {
 5:     arr = int[5];
 6:   }

 7:   public DebugMe(int[] arr, String n) {
 8:     arr = arr;
 9:   }

10:   public printMe() {
11:     for (i = 0; i < arr.length; i++) {
12:       System.out.println(name.length() + arr[i]);
13:     }
14:   }
15: }
```

BUG 1: 3: `private String name;`

BUG 2: 5: `arr = new int[5];`

BUG 3: 7: `public Ouch(int[] arr, String n)` or 7: `public void DebugMe(int[] arr, String n)`

BUG 4: 8: `this.arr = arr;`

BUG 5: 10: `public void printMe()`

BUG 6: 11: `for (int i = 0; i < arr.length; i++)`

BUG 7: 12: `name` is never initialized

Awesome TAs, What Are They Good For?

4. (15 points) The following program prints five lines of output. What are they? Write and circle your answer on the following page.

```
1: 4cheezburger4 4
2: 5fryz5 5
3: 4cheezburger4 14
4: 5fryz5 7
5: cheezburger fryz
```

QuickSelect

5. (20 points)
For each invocation of `QuickSelect` below, write exactly what the program will print:

(a) `% java QuickSelect 4 3 2 1 7 6 5`

```
0 7
return value = 4
1 3 2 4 6 5 7
```

(b) `% java QuickSelect 3 2 1 6 4 5 7`

```
0 7
3 7
return value = 4
1 2 3 4 7 6 5
```

Queues on First

6. (32 points)

(a) Implement the `atBat()` and `addToLineup` methods. Make sure that the methods obey the comments in the code extract above, and that `first`, `last`, and `size` are updated correctly.

```
public Player atBat {
  if (first) == null) return null;
  Person p = first;
  first = first.next;
  if (first == null) last = null;
  size--;
  return p;
}

public void addToLineup(Person p) {
  if (p == null) return;
  if (first == null) first = p;
  else last.next = p;
  p.next = null;
  last = p;
  size++;
}
```

(b) Implement the `eject()` method

```
public boolean eject(Player p) {
  if (first == null) return false;

  if (first == p) {
    first = first.next;
    if (first == null) last = null;
    size--;
    return true;
  }

  for (Player q = first; q.next != null; q = q.next) {
    if (q.next == p) {
      q.next = q.next.next;
      if (q.next == null) last = q;
      size--;
      return true;
    }
  }
}
```