Unit 3 Exam Key

Code Tracing - 2 points each

-1 For having quotes on any of the string arrays

#	Output	Notes
1	`[6, -2, 4]`	
2	`[8, 2, -10]`	-1 If they have another space at the end
3	`lines: 6` `numbers: 6`	

4	`[1, 6, 1, 0]`
	`[9, -4, 2, 10]`
	`[7, 11, -4, 0]`	

5	`[C, D, F, A]`	
6	`[home, not, just, anywhere, go, can, I]`	This output is on one line!
7	`18`	syntax error or runtime gets full points because they aren't familiar with not using braces for 1 line body of code
8	`-1133`	

OOP Code Tracing - 3 points each

#	Output	Notes
1	`false true`	
2	`300`	syntax error or runtime gets full points because pages should be in Printed

3	`oh my`
	`hey girl hey`
	`howdy`
		syntax error or runtime gets full points because of missing brace

PROGRAMMING

Program - OO Programming 1- 10 Points

```
4  public abstract class Ticket {
5      private int number;
6
7      public Ticket(int number) {
8          this.number = number;
9      }
10
11     public abstract double getPrice();
12
13     public String toString() {
14         return "Number: " + this.number +
15                 ", Price: " + this.getPrice();
16     }
17 }
```

Item	Line #	Item	Points (subtract if code not done or incorrect)	Notes
A	4	abstract class	-2 class is abstract	
B	7-8	constructor	-2 takes an int and assigns it to number	
C	11	getPrice()	-2 declared abstract	
D	13-16	toString()	-2 returns number with label -2 returns price with label	
X	N/A	Instructions	MAX of -5 -2 Unnecessary code -2 Redundant or inefficient code -2 Seriously incorrect style guide issue -2 Java syntax not allowed -2-4 Not following instructions (if not reflected above already)	

Program - OO Programming 2 - 25 Points

```
 3 public class Bear extends Critter {
 4     private int turnsToHibernate;
 5     private int turnsSoFar;
 6     private boolean hibernating;
 7
 8     public Bear(int turns) {
 9         turnsToHibernate = turns;
10     }
11
12     public boolean eat() {
13         boolean result = !hibernating;
14         hibernating = true;
15         return result;
16     }
17
18     public Attack fight(String opponent) {
19         return hibernating ? Attack.ROAR : Attack.SCRATCH;
20     }
21
22     public Direction getMove() {
23         Direction result = hibernating ? Direction.CENTER : Direction.NORTH;
24         if (hibernating) {
25             turnsSoFar++;
26             if (turnsSoFar == turnsToHibernate) {
27                 hibernating = false;
28                 turnsSoFar = 0;
29             }
30         }
31         return result;
32     }
33
34     public String toString() {
35         return hibernating ? "H" : "B";
36     }
37 }
```

Item	Line #	Item	Points (subtract if code not done or correct)	Notes
A	3	extends	-2 extend Critter	
B	4-6	fields	-1 int for turns to hibernate -1 int for turns so far -1 boolean for hibernating	
C	8-10	constructor	-1 sets hibernating to false -2 save turns to hibernate	
D	12-16	eat()	-1 sets hibernating to true -2 correct return value (had been hibernating returns false)	
E	18-20	fight()	-2 correct return value	
F	22-32	getMove()	-2 keeps track of turns so far -4 stops hibernating after turns to hibernate -4 correct return value	
G	34-36	toString()	-2 correct return value	
X	N/A	Instructions	MAX of -5 -2 Unnecessary code -2 Redundant or inefficient code -2 Seriously incorrect style guide issue -2 Java syntax not allowed	Don't count off if they have code for getColor() or not, or if it is correct or not.

			-2-4 Not following instructions (if not reflected above already)	

Program - ArrayLists - 20 Points

```
27      public static void modifyStringList(ArrayList<String> list) {
28          for (int i = 0; i < list.size(); i++) {
29              boolean remove = false;
30              String word = list.get(i).toLowerCase();
31              int j = 1;
32              while (j < word.length() && !remove) {
33                  if (word.charAt(j) == word.charAt(j - 1)) {
34                      remove = true;
35                      list.remove(i);
36                      i--;
37                  }
38                  j++;
39              }
40          }
41      }
```

Item	Line #	Item	Points	Notes
A	27	method header	-2 void return type -2 correct parameter	
B	28	loop	-2 loop to traverse ArrayList	
C	32-39	remove strings	-4 loop to traverse characters of one string -3 decision to remove the word -3 remove the word -2 stop processing string when first double letter is found -2 subtract 1 from counter if removed	
X	N/A	Instructions	MAX of -5 -2 Unnecessary code -2 Redundant or inefficient code -2 Seriously incorrect style guide issue -2 Java syntax not allowed -2-4 Not following instructions (if not reflected above already)	

Programming - Recursion - 20 Points

```
 9      public static void printRemoveDigit(int num, int digit) {
10
11          if (num < 0) {
12              System.out.print("-");
13              printRemoveDigit(-num, digit);
14          } else if (num < 10) {
15              if (num != digit)
16                  System.out.print(num);
17          } else {
18              printRemoveDigit(num / 10, digit);
19              printRemoveDigit(num % 10, digit);
20          }
21      }
```

Item	Line #	Item	Points	Notes
A	9	method header	-1 void return type -1 two integer parameters	
B	11-13	negative numbers	-1 check for negative number -1 print minus sign -2 call method for -num	
C	14-16	base case	-2 check if num < 10 -4 if num < 10 and num != digit to remove, print digit	
D	17-20	recursive case	-4 call method for num / 10 -4 call method for num % 10	Do NOT count off if they solved this with Strings.
X	N/A	Instructions	MAX of -5 -2 Unnecessary code -2 Redundant or inefficient code -2 Seriously incorrect style guide issue -2 Java syntax not allowed -2-4 Not following instructions (if not reflected above already)	