COS 126 Exam Review

• Exams overview
• Example programming exam
• Example written exam questions (part 1)
We have exams in the fall

- Two written exams.
- Two programming exams.
- Prep sessions in class meetings
- No exam "midterm week"
- No final exam

Programming exams.

- October 10 and December 5.
- Mini in-class assignments.
- “Can you write a short program?”
- You will need to practice.

Written exams.

- October 17 and December 12.
- Written by RS.  
  Note: RS has been known to re-use questions
- “Did you watch the lectures and do the reading?”
- One question per lecture (roughly).
See Exams tab for full details and old exams.

- Read carefully before each exam.
- Policies are the contract between us and you.

Policies (programming exam).
- Open course materials.
- No other web access.
- No outside communication.

Policies (written exam).
- Closed book/notes/computer.
- 1 page (one side) cheatsheet.
Things to remember about inclass exams

We know that you don't have much time.
- Exams are 50 minutes.
- "One page" programming exams.
- Five-minute questions on written exams.

We have to grade the exams.
- 400+ exams.
- No open-ended questions.
- Fully prepared rubrics.

Old exams are not completely reliable.
- Course offerings differ slightly.
- We have made mistakes in the past.
Programming Exam Logistics

Writing a short program in 50 minutes can be a challenge for anyone.
- You will use your own computer.
- You will download and edit a template.
- You will submit your solution in the same way as you do for assignments.

You don't all fit in this room.
- Pay attention and know where to go.
- Arrive early.
- Make sure your computer is charged.

Advice: Practice, practice, practice.
- Write some short programs on your own.
- Attend the practice programming exam.
- Try a past programming exam (untimed).
- Try another one (timed).
Q. Can you write a simple program on your own?

**Example** (Fall 2015).

**Part 1.** Write programs that find the number of distinct values among the integers on standard input, assuming that the input is nonempty and in sorted order.

**Your task.** Add code to the file Count1.java to print the number of integers on standard input and the number of distinct values among those integers.

**Details.** Write a single loop that uses StdIn.readInt() to read each integer one at a time, but do not save them in an array. To compute the number of distinct values, add code to the loop to update distinct if the new value just read differs from the value read just before it.

```java
public class Count1
{
    public static void main(String[] args)
    {
        int count = 1;
        int distinct = 1;
        // YOUR CODE HERE
    }
}
```

Exams Info gave instructions to load this before the exam

You also get a test file and desired output

% more testCount1tiny.txt
1 1 1 1 2 2 2 2 4 4 4 4 5 5 6 6 9 9
% java Count1 < testCount1tiny.txt
6 distinct values among 18 integers
Your task. Add code to print the number of integers on standard input and the number of distinct values among those integers.

% more testCount1tiny.txt
1 1 1 1 2 2 2 2 4 4 4 4 5 5 6 6 9 9
% java Count1 < testCount1tiny.txt
1 distinct values among 18 integers

% java Count1 < testCount1tiny.txt
6 distinct values among 18 integers

```java
public class Count1 {
    public static void main(String[] args) {
        int count = 1;    // number of integers
        int distinct = 1;  // number of different ones
        int val = StdIn.readInt();
        while (!StdIn.isEmpty()) {
            int newVal = StdIn.readInt();
            count++;          // count the integers
            if (newVal != val) {
                distinct++;     // count the different ones
                val = newVal;
            }
            StdOut.println(distinct + 
                            " distinct values among " 
                            + count + " integers");
        }
    }
}
```
Q. Can you quickly apply something you have recently learned?

**Example** (Fall 2015).

*Part 2.* Assume that the integers on standard input are nonnegative and less than a value $M$ given as the first command-line argument, *but not necessarily in order.*

**Details.** To compute the needed values, use two loops. First, write a loop that reads the integers one at a time from standard input, counts them, and uses the boolean array $b[\cdot]$ to record which values have been seen: when you read a value $v$, set $b[v]$ to true. Second, write a loop that counts the number of true values in $b[\cdot]$ (the number of distinct values in the input).

Q. Do you understand "coupon collector"?

```java
public class Count2 {
    public static void main(String[] args) {
        int M = Integer.parseInt(args[0]);
        boolean[] b = new boolean[M];
        // YOUR CODE HERE
    }
}
```

Exams Info gave instructions to load this before the exam

The description includes an example

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b[i]$</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
```java
public class Count2 {
    public static void main(String[] args) {
        int M = Integer.parseInt(args[0]); // no number higher than M - 1
        boolean[] b = new boolean[M]; // true means index was input

        // loop to read input, count integers and fill boolean array
        int count = 0;
        while (!StdIn.isEmpty()) {
            int val = StdIn.readInt();
            b[val] = true;
            count++;
        }

        // count distinct numbers
        int distinct = 0;
        for (int i = 0; i < M; i++)
            if (b[i]) distinct++;

        // output
        StdOut.println(distinct + " distinct values among " + count + " integers");
    }
}
```
Written Exam Logistics

The first exam is on Thursday Oct. 17.
- Prep session (first half) next.
- Prep session (second half) Tuesday Oct. 15.

You don't all fit in this room.
- Pay attention and know where to go.
- Arrive early.
- No calculator/phone/computer/headphones

Advice.
- Review lectures/reading.
- Try an old exam (untimed).
- Try another one (timed).
- Review a few more.
Example question: Basics

Q. Do you understand types and Java's type conversion and precedence rules?

Ex. (Fall 2014 Question 1) Give the type and value of each of the following Java expressions, supposing that it is used as the argument of a println() call.

(3 < 2) && (1 > 0)    boolean    FALSE
"800" + 99            String     80099
800 + 99 + "A"        String     899A
3 + (int) Math.random()  int        3
(double)(3 / 2)       double     1.0
3 / 2.0 + 2 / 5       double     1.5
(8 <= 2) || (2e88 <= 88e2) boolean    FALSE
Integer.parseInt("8.5*2")          ILLEGAL
8 / 2 * (2 + 2)        int        16

% jshell
| Welcome to JShell -- Version 11.0.2
| For an introduction type: /help intro

jshell> (3 < 2) && (1 > 0) $1 ==> false
jshell> "800" + 99 $2 ==> "80099"
jshell> 800 + 99 + "A" $3 ==> "899A"
jshell> 3 + (int) Math.random() $4 ==> 3
jshell> (double)(3 / 2) $5 ==> 1.0
Precedence matters

**The New York Times**

The Math Equation That Tried to Stump the Internet

Many respondents were certain the answer was 16. Others … insisted the right answer was 1. That’s when the trash talking began. “Some of y’all failed math and it shows,” said one. Another posted a photo showing that even two different electronic calculators disagreed. The normally reassuring world of math, where right and wrong exist, and logic must prevail, started to seem troublingly, perhaps tantalizingly, fluid.

“When arithmetic operators have the same precedence, the order is determined by left associativity, so that a - b - c and (a - b) - c represent the same sequence of operations.”

— Sedgewick and Wayne, p. 17

not "fluid" to a computer scientist!
Example question: Arrays

Q. Do you understand basic properties and rules about Java arrays?

**Ex.** (Fall 2016 Question 2) Which of the following statements are true for Java arrays? Mark each statement as either TRUE or FALSE.

- An array can't simultaneously store both an element of type double and an element of type boolean. **TRUE**
- Once you create an array, you cannot change its type. **TRUE**
- You must access the elements in an array in sequential order, e.g., you cannot access a[5] until you have accessed a[0], a[1], through a[4]. **FALSE**
- If a[] is a boolean array of length 126, then the expression a[i] will evaluate arbitrarily to either true or false if the index i is equal to 126. **FALSE**
- If a[] and b[] are two arrays of length 2, then a == b is true if and only if both a[0] == b[0] and a[1] == b[1] are true. **FALSE**
Example question: Loops and conditionals

Q. Can you figure out the effect of a simple Java program that uses `while` and `if` statements?

Ex. (Fall 2014 Question 2) Fill in the trace for just after each iteration of the outer `for` loop in this program:

```
int[] a = new int[N];
a[0] = 1;
for (int i = 1; i < N; i++) {
    int sum = 0;
    for (int j = 0; j < i; j++)
        sum = sum + a[j];
    a[i] = 1 + (2 * sum) / i;
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>sum</th>
<th>a[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>36</td>
<td>13</td>
</tr>
</tbody>
</table>

Write one line of code that could replace the body of the outer loop.

```
a[i] = 2*i + 1;
```
Example question: Input and output

Q. Do you understand basic ways of communicating with your programs?

Ex. (S2011 Q4) Give the results of invoking this program with the given commands.

```java
public class Q4 {
    public static void main(String[] args) {
        int curr = StdIn.readInt();
        StdOut.print(curr + " ");
        int prev = curr;
        while (!StdIn.isEmpty()) {
            curr = StdIn.readInt();
            StdOut.print((prev + curr) / 2 + " ");
            prev = curr;
        }
        StdOut.println();
    }
}
```

% more input.txt
```
2
4
6
8
10
12
8
2
```
% java Q4 < input.txt
```
2
3
5
7
9
11
10
5
```
% java Q4 < input.txt | java Q4
```
2
2
4
6
8
10
10
7
```

Note: It prints the first number, then the average of each number and its predecessor.
Mark your calendar

*Tuesday* October 8: Practice programming exam

*Thursday* October 10: PROGRAMMING EXAM

*Tuesday* October 15: Written exam prep (part 2)

*Thursday* October 17: WRITTEN EXAM