The ABCs of APCS-A

July 12, 2011
Welcome

We are:

- Renee Ciezki, College Board Advisor
  - *Glendale Community College, Glendale, AZ*
  - *Ironwood High School, retired May, 2011*

- Lester Wainwright, Question Writer Extraordinaire
  - *Charlottesville High School, Charlottesville VA*
Who are you?

Please stand if …

you are a high school teacher

Please remain standing if you have…

ever taught APCS-A

taught APCS-A for more than 5 years
Today’s Topics

APCS-A Course overview
Getting started
Resources
Gridworld case study
The exam
Best practices
Goals of AP

- Introductory course for computer science
- Course for other disciplines requiring technology
- College credit
- Experience challenging course
Reaching and Teaching ALL Students
Introductory Programming Environments

ALICE - http://www.alice.org/

GREENFOOT - http://www.greenfoot.org/

JELIOT - http://cs.joensuu.fi/jeliot/

RACKET – Program By Design

SCRATCH - http://scratch.mit.edu/
Students with Special Needs
(Physically or Otherwise Challenged)
Students with Different Personalities and Learning Styles
Students Who Challenge Your Knowledge

Will Teachers Learn?

Greg Smith

I read about it on the www.techedknow.com blog. You should subscribe, so you can learn about technology too.

You what on the what?

When will these teachers ever learn!?!
Students Living in a Technological Age
Getting Started – Resources for APCS - A

• APCS Home Page

• Summer Institute
  ◦ http://apcentral.collegeboard.com/apc/Pageflows/InstitutesAndWorkshops/InstitutesAndWorkshopsController.jpf

• AP Audit

• University faculty - local or worldwide help from the Electronic Discussion Group.
Additional Resources for CS AP A

• Codingbat – practice writing Java or Python code
  o http://codingbat.com/
• CSTA – curriculum, online repository
  o http://csta.villanova.edu/handle/2378/1
• CS Unplugged – lessons and activities without a computer
  o http://www.csunplugged.org
• Exploring CS – curriculum, other resources
  o http://www.exploringcs.org/
• Lightbot – a fun, easy introduction to programming
  o http://www.kongregate.com/games/Coolio_Niato/light-bot
Classroom Resources

Materials from other teachers

- Barb Ericson – practice multiple-choice
  - http://manatee.cc.gt.atl.ga.us/apExam/
- Roger Frank – lab ideas
  - http://www.rfrank.net/cslabs-final/cslabs.html
- Michael Lew – videos and projects
  - http://thecubscientist.com/APCS/indexAPCS.html
- Glen Martin – GridWorld projects
  - http://www.martin.apluscomputerscience.com/gridworld.html

Review books – Barron’s, Litvins’ Be Prepared
Questions and Comments?
Developing the AP CS Course and Exam

Three major players
• College Board
• ETS
• Development Committee
The Course

Course Description revised as needed
- Java updates
- Subset adjustments
- Curriculum surveys

Case Study
- GridWorld
- Several multiple-choice questions
- One free-response question
Why Case Studies?

- Allow students the benefits of apprenticeship.
- Make large programs accessible to students.
- Well suited for active learning and team work.
- Encourage students to reflect on solutions.
Examination: Multiple Choice

• 40 questions
  o Draft questions written by committee members and instructors
  o Questions finalized by committee
  o Questions pre-tested by colleges

• Equater Set ensures consistency in scoring

• No penalty for guessing
Examination: Free Response

- 4 Questions
- A variety of topics
- Meaningful questions that can be answered in a short amount of time
- Currently writing the 2013 exam
- Green sheets being discontinued
Free Response Scoring
Exam Questions & Scoring Guidelines

Resources available on AP Central
  - Free-Response Questions
  - Scoring Guidelines
  - Student Performance Q & A
  - Scoring Statistics
  - Sample Student Responses & Scoring

# Free Response Scoring

## Exam Questions & Scoring Guidelines

|-------------------------|------|------|------|------|------|------|------|------|------|----------|

### 2011: Free-Response Questions

<table>
<thead>
<tr>
<th>Questions</th>
<th>Scoring</th>
<th>Samples and Commentary</th>
<th>Score Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Questions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2010: Free-Response Questions

<table>
<thead>
<tr>
<th>Questions</th>
<th>Scoring</th>
<th>Samples and Commentary</th>
<th>Score Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Questions</td>
<td>Scoring Guidelines</td>
<td>Sample Responses Q1</td>
<td>Score Distributions</td>
</tr>
<tr>
<td></td>
<td>Student Performance Q&amp;A</td>
<td>Sample Responses Q2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scoring Statistics</td>
<td>Sample Responses Q3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sample Responses Q4</td>
<td></td>
</tr>
</tbody>
</table>
2011 AP® COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

COMPUTER SCIENCE A
SECTIONS II
Time—1 hour and 45 minutes
Number of questions—4
Percent of total score—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:
• Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
• Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
• In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

1. Digital sounds can be represented as an array of integer values. For this question, you will write two unrelated methods of the Sound class.

A partial declaration of the sound class is shown below.

```java
public class sound {
    // The array of values in this sound; guaranteed not to be null */
    private int[] samples;

    // Changes these values in this sound that have an amplitude greater than limit.
    // Values greater than limit are changed to limit.
    // Values less than -limit are changed to -limit.
    // @param limit the amplitude limit
    // @precondition limit \geq 0
    // @return the number of values in this sound that this method changed
    public int limitAmplitude(int limit) {
        // to be implemented in part (a) */
    }

    // Removes all silence from the beginning of this sound.
    // Silence is represented by a value of 0.
    // Postcondition: samples contains at least one nonzero value
    // @postcondition: the length of samples reflects the removal of starting silence
    public void trimSilenceFromBeginning() {
        // to be implemented in part (b) */
    }
}
```

There may be instance variables, constructors, and methods that are not shown.

AP® Computer Science A is a registered trademark of the College Board. All other products and services may be trademarks of their respective owners. Visit the College Board on the Web: www.collegeboard.org. Permission to copy this material is granted to educators and students to reproduce for educational and noncommercial purposes.
Part (a)  getTotalBoxes  3 points

**Intent:** Compute and return the sum of the number of boxes of all cookie orders in this.orders

+1  Considers all CookieOrder objects in this.orders
+1/2  Accesses any element of this.orders
+1/2  Accesses all elements of this.orders with no out-of-bounds access potential

+1  Computes total number of boxes
+1/2  Creates an accumulator (declare and initialize)
+1/2  Invokes getNumBoxes on object of type CookieOrder
+1/2  Correctly accumulates total number of boxes

+1/2  Returns computed total

Part (b)  removeVariety  6 points

**Intent:** Remove all CookieOrder objects from this.orders whose variety matches cookieVar; return total number of boxes removed

+4  Identifies and removes matching CookieOrder objects
+1/2  Accesses an element of this.orders
+1/2  Compares parameter cookieVar with getVariety() of a CookieOrder object (must use .equals or .compareTo)
+1  Compares parameter cookieVar with getVariety() of all CookieOrder objects in this.orders, no out-of-bounds access potential
+1  Removes an element from this.orders
+1  Removes only matching CookieOrder objects
+1  Removes all matching CookieOrder objects, no elements skipped

+1  Computes total number of boxes in removed CookieOrder objects
+1/2  Creates an accumulator (declare and initialize)
+1/2  Invokes getNumBoxes on object of type CookieOrder
+1/2  Correctly accumulates total number of boxes

+1/2  Returns computed total

**Usage:**
-1  consistently references incorrect name instead of orders, of potentially correct type (e.g., this, MasterOrder)
-1  consistently references incorrect name instead of orders, incorrect type
(a) The `getTotalBoxes` method computes and returns the sum of the number of boxes of all cookie orders. If there are no cookie orders in the master order, the method returns 0.

Complete method `getTotalBoxes` below.

```java
/** @return the sum of the number of boxes of all of the cookie orders */
public int getTotalBoxes()
{
    int sum = 0;
    for (int k = 0; k < orders.length; k++)
        sum += orders[k];
    return sum;
}
```

---

**AP® COMPUTER SCIENCE A**

**2010 SCORING COMMENTARY**

**Question 1**

**Overview**

This question focused on the `ArrayList` data structure, element access and removal, algorithms that required processing all elements, and using instance data. Students were provided with the frameworks for two classes, `CookieOrder` and `MasterOrder`, and were asked to implement two methods in the `MasterOrder` class. In part (a), students were required to implement the method `getTotalBoxes()` that returns the sum of the number of boxes of all of the cookie orders in the `ArrayList` instance variable. This could be accomplished by invoking `getNumBoxes()` on each element of the list, accumulating and returning the sum. In part (b), students were required to implement the `removeVariety()` method, which removes from the `ArrayList` instance variable all `CookieOrder` objects that have the same variety as the parameter, maintains an accumulator of the number of boxes removed, and returns the accumulator's final value. This could be accomplished by first invoking `getVariety()` on each element of the list and performing a string comparison with the parameter. If the two strings match, the result of invoking `getNumBoxes()` would be added to an accumulator and the `remove()` method invoked to delete that order from the list. The accumulated total needed to be returned at the end of the method.

**Sample: 1A**

**Score: 9**

In part (a), the initial check for a zero-length list is unnecessary, but it does not cause a problem with the solution. The student correctly declares and initializes an accumulator. The student then correctly uses an indexed for-loop to access every element of the `ArrayList`, calls `getNumBoxes()` on each element and accumulates the sum. When the loop ends, the sum is returned. Part (a) earned all 3 points.

In part (b), the student correctly declares and initializes an accumulator and uses a descending indexed for-loop to access every element of the `ArrayList`. The student correctly invokes `getVariety()` on each `CookieOrder` and compares the result with the `cookieVar` parameter. If they match, the student invokes `getNumBoxes()` and adds the result to the accumulator. The student then uses `remove()` to delete that cookie order from the list. After the loop, the student returns the accumulated total number of boxes removed. Part (b) earned all 6 points.

**Sample: 1B**

**Score: 6**

In part (a), the initial check for a zero-length list is unnecessary, but it does not cause a problem with the solution. The student correctly declares and initializes an accumulator. The student then correctly uses an indexed for-loop to access every element of the `ArrayList`. However, the student does not call `getNumBoxes()` on each element, and the accumulator is not counting boxes, so those two ½ points were not earned. A computed total is returned, earning ½ point. Part (a) earned 2 points.

In part (b), the reference `orders[1]` earned ½ point for “Accesses an element of this orders.” The call to `getVariety()` and the comparison to the `cookieVar` parameter are done correctly. The student uses `orders.removeOrder(1)` instead of `orders.remove()`, so did not earn the ½ point for “Removes an element.” The attempted removal is appropriately guarded and earned the ½ point for “Removes only matching `CookieOrder` objects.” The student uses an ascending index for-loop without
AP CS A Exam

Link to EDG (http://lyris.collegeboard.com/read/?forum=ap-compsci&s=1 -- see recent AP exam scoring ?? thread) for readers’ comments and recommendations.