How am I Going to Grade All These Assignments?

Thinking About Rubrics in the Large

John Cigas

Crystal Furman

Adrienne Decker

Tim Gallagher

AP Program Goal

- AP gives students the chance to tackle college-level work while they’re still in high school and earn college credit and placement.

Goals for Scoring

- Consistent and reliable application of rubric
- … across all readers
- … from start to finish

- A student’s grade should not be dependent on whether it was read day 1 or day 7.

Creating Questions and Drafting Rubric

Adrienne Decker

Associate Professor, Rochester Institute of Technology

AP Reader, AP Question Leader, AP Development Committee Member

Quizzes, Exams, and Assessments – Oh my!

- First – decide what topics or skills you want to assess
- Second – develop questions that ask students to demonstrate those skills
- Third – develop a rubric that will assign values appropriately to demonstrate varying levels of knowledge by student
- Fourth – apply rubric to student responses
- Fifth…

What do we want to assess?

- We want to assess student knowledge of 2D arrays
    - Traversal
    - Information Retrieval
    - Creation
    - Storing information into the structure
    - 2D array that holds objects (not just primitives)

Example Question

- **This question asks you to write methods that process a 2D integer array that contains consecutive values. Each of these integers may be in any position in the 2D integer array. For example, the following 2D integer array with 3 rows and 4 columns contains the integers 5 through 16, inclusive.**

	0	1	2	3
0	15	5	9	10
1	12	16	11	6
2	14	8	13	7

Example Question

- **The following `Position` class is used to represent positions in the integer array. The notation `(r,c)` will be used to refer to a `Position` object with row `r` and column `c`.**

```
public class Position
{
   /** Constructs a Position object with row r and column c. */
  public Position (int r, int c)
  {    /* implementation not shown */  }

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

Part a

- Write a `static` method `findPosition` that takes an integer value and a 2D integer array and returns the position of the integer in the given 2D integer array. If the integer is not an element of the 2D integer array, the method returns `null`.

-
```
public static Position findPosition
                            (int num, int[][] intArr)
```

Part b

- The *successor* of an integer value is the integer that is one greater than that value. A 2D *successor array* shows the position of the successor of each element in a given 2D integer array. Each element in the 2D successor array is the position (row, column) of the corresponding 2D integer array element's successor. The largest element in the 2D integer array does not have a successor in the 2D integer array, so its corresponding position in the 2D successor array is `null`.

-
  ```
  public static Position[][]
      getSuccessorArray(int[][] intArr)
  ```

Part b

arr	0	1	2	3
0	15	5	9	10
1	12	16	11	6
2	14	8	13	7

succs	0	1	2	3
0	`(1,1)`	`(1,3)`	`(0,3)`	`(1,2)`
1	`(2,2)`	`null`	`(1,0)`	`(2,3)`
2	`(0,0)`	`(0,2)`	`(2,0)`	`(2,1)`

Steps 1 & 2 complete! On to grading!

- What parts of the solution are critically important?
- What do we want to assign points for?

Welcome to the Audience Participation Part of our Program!

What should we assign points to?

So, here were our ideas (Part a)

- Traverse 2D array appropriately (don’t mess up bounds, don’t skip elements)
- Find element equal to parameter
- Construct Position with correct row and column of found value
- Return the correct Position
- Be able to identify that value not in 2D array and return null

So, here were our ideas (Part b)

- Creates new 2D array of Position objects with correct size
- Find a successive position based on a value in the array
- Put that position into the new array
- Do that for all the values in the original array and return result

Formalizing the Rubric

John Cigas

Professor, Park University

AP Chief Reader

Part a – Assessment Rubric

Part (a) `findPosition` 5 pts

+1	Accesses all necessary elements of `intArr` (*no bounds errors*)
+1	Identifies `intArr` element equal to `num` (*in context of an* `intArr` *traversal*)
+1	Constructs `Position` object with same row and column as identified `intArr` element
+1	Selects constructed object when `intArr` element identified; `null` when not
+1	Returns selected value

Part b – Assessment Rubric

Part (b) **`getSuccessorArray`** **4 pts**

+1 Creates 2D array of Position objects with same dimensions as `intArr`

+1 Assigns a value to a location in 2D successor array using a valid call to `findPosition`

+1 Determines the successor `Position` of an `intArr` element accessed by row and column (*in context of an* `intArr` *traversal*)

+1 Assigns all necessary locations in successor array with corresponding position object or null (*no bounds errors*)

The return from this method is not being assessed.

Question-Specific Penalties

-1	Uses confused identifier `Arr`
-1	Uses `intArr[].length` as the number of columns
-1	Uses non-existent accessor methods from `Position`

General Scoring Guidelines (1-point penalties)

v) Array/collection access confusion `([] get)`

w) Extraneous code that causes side-effect *(e.g., writing to output, failure to compile)*

x) Local variables used but none declared

y) Destruction of persistent data *(e.g., changing value referenced by parameter)*

z) Void method or constructor that returns a value

General Scoring Guidelines (No penalty)

- Extraneous code that causes no side-effect
- Spelling/case discrepancies where there is no ambiguity
- Local variable not declared provided other variables are declared in some part
- Common mathematical symbols used for operators (• ÷ <> ≠)
- **`[]` vs. `()` vs. <>**
- `=` instead of `==` and vice versa
- **`length/size` confusion for array, String, List, or ArrayList; with or without ( )**
- Extraneous `[]` when referencing entire array
- `private` or `public` qualifier on a local variable
- **Extraneous size in array declaration,**
  - ***e.g.* `int[size] nums = new int[size];`**
- Missing `;` provided line breaks and indentation clearly convey intent
- Missing { } where indentation clearly conveys intent
- Missing ( ) around `if` or `while` conditions

Samples

- Try out rubric on samples of student work
- Use a small group of people (1-3)
- Look for student work that differs from canonical solution
    - Oversights
    - Missing pieces
    - Valid, but different approach
    - Incorrect algorithm/approach
    - Very little code

Oversights

- Syntax – {}, (), [], ;
- Missspellings
- Incorrect execution
  - if (0 < x < 10)
  - input(int(“Enter a score: “))
- Missing return
- Missing declaration or initialization
- Off by one

Missing Pieces

- What happens when something isn’t there?
- Probably don’t want a ripple effect
- Limited what-if
    - For each element in the canonical solution
        - Delete that element
        - Reapply the rubric

Canonical Solution

```
for (int r = 0; r < intArr.length; r++)
   for (int c = 0; c < intArr[0].length; c++)
      if (intArr[r][c] == num)
         return new Position(r, c);
return null;
```

Incorrect algorithm

- How to score this?

```
for (int r = 0; r < intArr.length; r++)
    for (int c = 0; c < intArr[0].length; c++)
        if (intArr[r][c] == num)
            return new Position(r, c);
        else
            return null;
```

Annotate rubric with examples

Rubric Criteria	Responses will not earn the point if they...
Accesses all necessary elements of intArr (no bounds errors)	● use if (...) return; else return null; inside loop ● confuse row and column bounds ● fail to traverse intArr
Identifies intArr element equal to num (in context of an intArr traversal)	● use .equals instead of ==
Constructs Position object with same row and column as identified intArr element	● omit keyword new ● use (r,c) instead of Position(r,c)
Selects constructed object when intArr element identified; null when not	● use if (...) return; else return null; inside loop ● use (r,c) instead of Position(r,c)

Different approaches

- Ask for recursion, get a loop
- Ask for one algorithm, get a different one
- Expect parsing a string character by character, but get 13 calls to various, valid string functions instead.

Not much code

- Be comfortable with scores
- Give some credit
- Don’t give too much

End product

- More precise wording on rubrics
- Specific examples of how to apply rubric in different situation

Training and Consistency

Tim Gallagher

Computer Science Teacher, Winter Springs High School

AP Teacher, AP Question Leader, AP Exam Leader

Training to Use the Rubric

- Scoring each point individually.
- Don’t try to grade/look at the entire problem at once.
    - Avoid “This paper FEELS like a 5/10…”
- Keep deductions on the rubric first, if possible. Use the “General Scoring Guidelines” only if the deduction doesn’t fit on the rubric.
- Once a point is lost, continue grading “as if” it was originally correct. No multiple deductions for a single error.
- Time spent on consistency and accuracy during training yields big dividends later.

Consistency is Key!

- Each point has been developed to score the same way across multiple students’ exams/projects.
- Score for accuracy – not speed!
    - With accuracy and a solid rubric, speed naturally develops over time.
- Grade samples first.
    - If using the same assignment or project from a previous year, train with some preset examples of what you know to expect from students as common solutions/mistakes.

Audience Participation

- How would you score these?

Part a - How Does This Score?

```
public static Position findPosition
                    (int num, int[][] intArr)
{
  Position index;
  for (int row=0; row < intArr.length; row++)
  {
     for (int col=0; col < intArr[0].length; col++)
     {
      if (intArr[row][col] == num)
            index = new Position(row,col);
      else index = null;
     }
  }
  return index;
}
```

?	?	?	?	?

Part a - Too Many Selections

```
public static Position findPosition
                (int num, int[][] intArr)
{
  Position index;
  for (int row=0; row < intArr.length; row++)
  {
     for (int col=0; col < intArr[0].length; col++)
     {
      if (intArr[row][col] == num)
           index = new Position(row,col);
      else index = null;
     }
  }
  return index;
}
```

1	1	1	0	1

Part b - How Does This Score?

```
public static Position[][]
   getSuccessorArray(int[][] intArr)
{
  int[][] newArr =
     new Position[intArr.length][intArr[0].length];

  for (int row=0; row < intArr.length; row++)
  {
     for (int col=0; col < intArr[0].length; col++)
     {
        newArr[row][col] =
            findPosition(intArr[row][col]+1, intArr);
     }
  }
  return newArr;
}
```

?	?	?	?

Part b - Successor Array Typing

```
public static Position[][]
   getSuccessorArray(int[][] intArr)
{
   int[][] newArr =
      new Position[intArr.length][intArr[0].length];

   for (int row=0; row < intArr.length; row++)
   {
      for (int col=0; col < intArr[0].length; col++)
      {
         newArr[row][col] =
            findPosition(intArr[row][col]+1, intArr);
      }
   }
   return newArr;
}
```

0	1	1	1

Have TAs Grade in pairs

- To begin, have two TAs split a small portion of papers, each grade half, then switch and grade the other half.
  - Reconcile differences
  - Time spent here eliminates confusion and inaccuracies later
- While grading, it always helps to have a partner to help clarify issues.
- While questions are always good, avoid going to far down “What if” scenarios.

Feedback for your TA Graders

- Calibrate and check for consistency between graders.
- Check in periodically to help graders gain confidence and to provide feedback.
- Positive feedback is just as important as corrections.
- Graders generally don’t like the “No news is good news” approach.
- People want to hear when they are doing well, as well as when they need critical feedback or corrections.

Questions/Comments

- Now it’s your turn!

Become a Reader!

- https://apcentral.collegeboard.org/professional-development/become-an-ap-reader