Be Prepared

for the

AP

Computer Science
Exam in Java

Chapter 5: Annotated Solutions
to Past Free-Response Questions

2017

Maria Litvin
Phillips Academy, Andover, Massachusetts

Gary Litvin
Skylight Publishing, Andover, Massachusetts

The free-response questions for this exam are posted on apstudent.collegeboard.org and, for teachers, on AP Central:

- For students: apstudent.collegeboard.org

- For teachers: apcentral.collegeboard.org/courses

Scoring guidelines are usually posted over the summer.

The www.skylit.com/beprepared/x2017all.zip file contains complete Java classes that include solutions and test programs for runnable projects.

Question 1

Part (a)

```
public Digits(int num)
{
  digitList = new ArrayList<Integer>();

  while (num > 0)
  {
    digitList.add(0, num % 10);  1, 2
    num /= 10;
  }

  if (digitList.size() == 0)  3
    digitList.add(0);  4
}  5
```

Notes:

1. Since digits in `num` are obtained from right to left, you have to add them at the beginning of `digitList`.

2. This is OK due to autoboxing; otherwise it would be

   ```
   digitList.add(0, new Integer(num % 10));
   ```

3. Don't forget the special case `num = 0`.

4. Or, without autoboxing:

   ```
   digitList.add(new Integer(0));
   ```

5. You can take care of `num = 0` first:

```
public Digits(int num)
{
  digitList = new ArrayList<Integer>();

  if (num == 0)
  {
    digitList.add(0);
    return;
  }

  while (num > 0)
  {
    digitList.add(0, num % 10);
    num /= 10;
  }
}
```

Or use a do-while loop to avoid handling 0 separately altogether:

```
public Digits(int num)
{
  digitList = new ArrayList<Integer>();

  do
  {
    digitList.add(0, num % 10);
    num /= 10;
  } while (num > 0);
}
```

Part (b)

```
public boolean isStrictlyIncreasing()
{
  for (int k = 1; k < digitList.size(); k++)
    if (digitList.get(k).intValue() <=
                          digitList.get(k-1).intValue()) [1]
      return false;

  return true;
}
```

Notes:

1. Or, simply,

```
    if (digitList.get(k) <= digitList.get(k-1))
```

due to unboxing.

Question 2

```java
public class MultPractice implements StudyPractice
{
  private int first, second;

  public MultPractice(int num1, int num2)
  {
    first = num1;
    second = num2;
  } ¹

  public String getProblem()
  {
    return first + " TIMES " + second;
  }

  public void nextProblem()
  {
    second++;
  }
}
```

Notes:

1. Or:

```java
public MultPractice(int first, int second)
{
  this.first = first;
  this.second = second;
}
```

Question 3

Part (a)

```
public void replaceNthOccurrence(String str, int n, String repl)
{
  int i = findNthOccurrence(str, n);
  if (i >= 0)
    currentPhrase = currentPhrase.substring(0, i) + repl +
                    currentPhrase.substring(i + str.length());
}
```

Part (b)

```
public int findLastOccurrence(String str)
{
  int i = -1, i2 = 0;
  int n = 1;

  while (i2 >= 0)
  {
    i2 = findNthOccurrence(str, n);
    if (i2 >= 0)
      i = i2;
    n++;
  }
  return i;
} [1, 2]
```

Notes:

1. The question description says (top of Page 15): "You must use findNthOccurrence appropriately to receive full credit." So just

   ```
   return currentPhrase.lastIndexOf(str);
   ```

   won't receive full credit. This is why it is important to always read carefully the entire question. It is also important to quickly make the safest guess of the real intent of the question, rather than exploiting possible ambiguities. For example, what does "appropriately" really mean here? Will the following solution receive full credit?

   ```
   public int findLastOccurrence(String str)
   {
     for (int i = currentPhrase.length() - str.length(); i >= 0; i--)
       if ((new Phrase(currentPhrase.substring(i))).
                                        findNthOccurrence(str, 1) == 0)
         return i;
     return -1;
   }
   ```

   We believe it will, but we will know for sure after the exams have been graded.
2. Alternative solutions, starting at high n and going down:

```
public int findLastOccurrence(String str)
{
  for (int n = currentPhrase.length(); n > 0; n--)
  {
    int i = findNthOccurrence(str, n);
    if (i >= 0)
      return i;
  }
  return -1;
}
```

Or:

```
public int findLastOccurrence(String str)
{
  int n = currentPhrase.length(), i = -1;
  while (n > 0 && i == -1)
  {
    i = findNthOccurrence(str, n);
    n--;
  }
  return i;
}
```

Question 4

Part (a)

```
public static Position findPosition(int num, int[][] intArr)
{
  for (int r = 0; r < intArr.length; r++)
    for (int c = 0; c < intArr[0].length; c++) [1]
      if (intArr[r][c] == num)
        return new Position(r, c);
  return null;
}
```

Notes:

1. Only rectangular arrays are in the AP subset, and the precondition guarantees that `intArr` has at least one row.

Part (b)

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

  Position[][] succ = new Position[rows][cols];

  for (int r = 0; r < rows; r++)
  {
    for (int c = 0; c < cols; c++)
    {
      int num = intArr[r][c];
      succ[r][c] = findPosition(num + 1, intArr);
    }
  }
  return succ;
}
```