COMPSCI 105: Principles of Computer Science
Summer 2007

Tutorial One: Nested Loops and Multi--Dimensional Arrays

Helen Gu and Slobodan Vukanovic
{ygu029, svuk002}@ec.auckland.ac.nz

This tutorial is not being assessed. It provides you with an opportunity to become familiar with concepts introduced in lectures.

Question One: One--Dimensional Arrays

(a) Use the space provided to write the output of the code below.

```
class QuestionOne {

      public static void main (String[] args) {
         int[] a = {1, 2, 3, 4, 1024, 5, 6, 7, 8, 9, 10};

         for (int i = 0; i < a.length - 1; i++)
            if (a[i] > a[i+1])
               swap (a, i, i+1);
         for (int i = 0; i < a.length; i++)
            System.out.print (a[i] + " ");
         System.out.println ();
      }


      public static void swap (int[] a, int i, int j) {
         int temp = a[i];
         a[i] = a[j];
         a[j] = temp;
      }

}
```

The output is: 1 2 3 4 5 6 7 8 9 10 1024

(b) Use the space below to write a routine (method) which prints the average word length in a given string array.

```
void printAverage (String[] words) {

      double partialSum = 0.0;

      for (int i = 0; i < words.length; i++)
         partialSum += words[i].length;

      System.out.println (partialSum / words.length);
}
```

Question Two: Multi--Dimensional Arrays

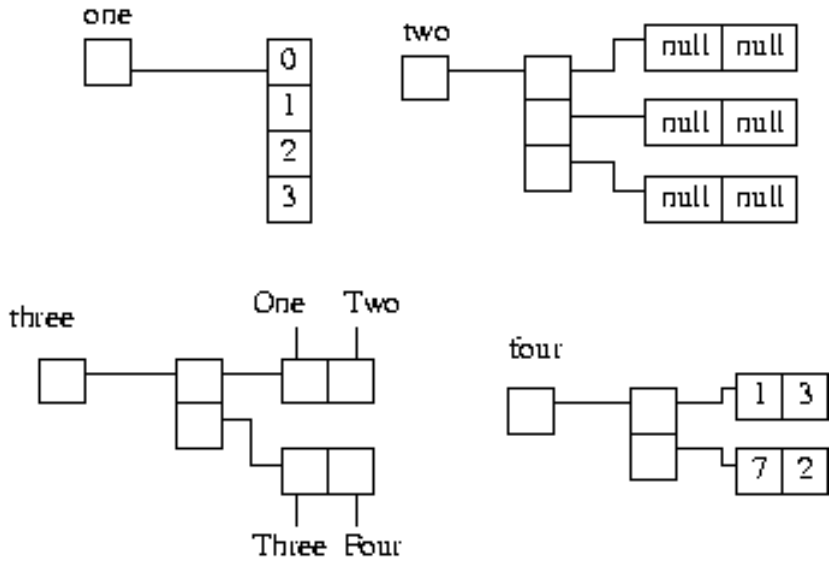
(a) How would we visualize the following four arrays after they have been created and initialized? Draw the diagrams in the box below the code.

```
int[] one = {0, 1, 2, 3};

Integer[][] two = new Integer[3][2];

String[][] three = {{"One", "Two"}, {"Three", "Four"}}

int[][] four = new int[2][2];
for (int i = 0; i < four.length; i++)
   for (int j = 0; j < four[i].length; j++)
      four[i][j] = (int) (Math.random () * 10);
```



Question Three: Nested Loops

(a) Use the space provided to write the output of the code below.

```
class Blaise {

     public static void main (String[] args) {
        blaise (4);
     }

     public static void blaise (int num) {
        System.out.println ("1");

        int[] prev = {1, 1};
        for (int i = 0; i < prev.length; i++)
           System.out.print (prev[i] + " ");
        System.out.println ();

        for (int i = 0; i < num; i++) {
           int[] next = new int[prev.length+1];
           for (int j = 0; j < next.length; j++) {
              if (j - 1 >= 0 && j < next.length - 1)
                 next[j] = prev[j-1] + prev[j];
              else
                 next[j] = 1;
              System.out.print (next[j] + " ");
           }
           prev = next;
           System.out.println ();
        }

     }
}
```

The output is:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

(b) Use the space provided to complete the function (method) below that takes an array of words and returns the most frequently occurring character. Assume that the strings are in lower case and that they contain only alphabetic characters. The idea is to keep the counts of occurrences of each individual character in the `freqs` array. We can then simply iterate through the characters of each word, and update the counts of those characters in `freqs`. Note that the `String` class in Java defines a `charAt (int i)` instance method, which returns the character at position `i` in the string.

```
public static char mostFrequent (String[] words) {
   int[] freqs = new int[26];

   for ( int i = 0 ; i < words.length; i++) {
      String currWord = words[i];
      for (int j = 0; j < currWord.length (); j++) {
         char currChar = currWord.charAt (j);
         freqs[currChar - 'a'] += 1;
      }
   }

   int max = 0;
   int indexMostFrequent = 0;
   for (int i = 0; i < freqs.length; i++)
      if (freqs[i] > max) {
         max = freqs[i];
         indexMostFrequent = i;
      }

    return (char) ('a' + indexMostFrequent);
}
```

Question Four: Nested Loops and Multi--Dimensional Arrays

(a) The classic matrix multiplication algorithm for square matrices of the same size comes straight from the definition:

> Let A and B be two n × n matrices to be multiplied, and let C be their product. Each entry in C is calculated as:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

Note that, if A and B are n × n, then their product C will also be n × n.

Your task is to download `MMul.java` from the course website and convert the pseudo--code given in the file to Java code. There are two test cases provided in `MMul.java` that you can use to test your code.