Midterm 3 Solution

CS 1323, Spring 2015

Name (printed legibly):

Section (1 meets at 9:30, 10 meets at 1:30):

Student number:

**Integrity Pledge**

**On my honor, I affirm that I have neither given nor received inappropriate aid in the completion of this exercise.**

**Signature:**

Do not write anything on the back of any page. If you need additional space, use the extra blank page at the back of the examination.

Answer all programming questions in Java.

Unless otherwise indicated, each part of a problem is worth the same number of points.

Show your work to receive partial credit.

Pay attention to whether you are asked for a code fragment, a method, or a complete program. Students who write complete programs for questions that requested code fragments are unlikely to be able to complete the examination during the assigned time.

You do not need to import packages or throw exceptions on this examination.

**This examination focuses on arrays, not ArrayLists. Answers that use ArrayList objects instead of arrays will not receive credit.**

1. (10 points; 2 points each for a) and b), 6 points for c))

On Friday night, the OU Sooners Mens’ Gymnastics Team won the national championship (and it was AWESOME).

In gymnastics, each team fields six competitors per event. Each competitor’s score is rated from 0.0 to around 20.0 on each event.

A team’s score on an event is the sum of the scores of the six competitors. High scores are better than low scores.

a) Declare a reference for an array that will store the scores for the competitors for one event. Do not construct the array here.

double[] scores;

b) Construct the array that you declared in a).

scores = new double[6];

c) Two perfect sized arrays, with references ou and pennState, already hold the scores on a single event for these two teams. Write a code fragment that prints out “OU won” or “Penn State won”, depending on scores stored in the two arrays.

double ouScore = 0.0;

double pennScore = 0.0;

// Add up the scores

for(int index=0; index<ou.length; ++index)

{

pennScore += penn[index];

ouScore += ou[index];

}

// Make the comparison

if (ouScore > pennScore)

{

System.out.println(“OU won!”);

}

else

{

System.out.println(“Penn State won!”);

}

1. (15 points; 9 points for a), 6 points for b))

a) Trace the code below using a memory diagram. You do not need to show variable i in your diagram. Show the array indices and array’s length field for all arrays in the heap.

public class Example // Line 1

{

public static void main(String[] args) // Line 2

{

double[] sample = {2.1, 3.2, 4.3, 5.4}; // Line 3

double[] result = new double[sample.length]; // Line 4

mystery(sample, result); // Line 5

}

public static double[] mystery(double[] source, double[] destination) // Line 6

{

destination = new double[source.length]; // Line 7

for (int i=0; i<source.length; ++i) // Line 8

{

destination[source.length-i-1] = source[i]; // Line 9

}

return destination; // Line 10

}

}

|  |  |  |
| --- | --- | --- |
|  | heap | |
|  | **Address** | **Contents** |
| 0 | 1000 | 2.1 |
| 1 | 1001 | 3.2 |
| 2 | 1002 | 4.3 |
| 3 | 1003 | 5.4 |
| length | 1004 | 4 |
| 0 | 1005 |  |
| 1 | 1006 |  |
| 2 | 1007 |  |
| 3 | 1008 |  |
| length | 1009 |  |
|  | 1010 |  |
| 0 | 1011 | 5.4 |
| 1 | 1012 | 4.3 |
| 2 | 1013 | 3.2 |
| 3 | 1014 | 2.1 |
| length | 1015 | 4 |
|  | 1016 |  |

|  |  |  |
| --- | --- | --- |
|  | main stack frame | |
|  | **Address** | **Contents** |
| sample | 100 | 1000 |
| result | 101 | 1005 |
|  | 102 |  |
|  | 103 |  |
|  | 104 |  |
|  | 105 |  |
|  | 106 |  |

|  |  |  |
| --- | --- | --- |
|  | mystery stack frame | |
|  | **Address** | **Contents** |
| source | 200 | 1000 |
| destination | 201 | ~~1005~~ 1011 |
|  | 202 |  |
|  | 203 |  |

b) The method above contains three mistakes that a memory diagram can help you find**. Find two of these mistakes** and put the line numbers below.

Mistake #1: Line 4: Sample array was constructed when it should not have been.

Mistake #2: Line 5: Returned array reference was not stored in main program.

Mistake #3: Line 6: Array reference destination should not have been passed to the method.

Mistake #4: Line 7: Array passed by sharing was reconstructed.

1. (15 points; 5 points for a), 10 points for b)) For each of the loops below, **find the value of sum at the end of the loop**. You may do this by tracing the loop using a table (at the right), or by briefly explaining what the code does and giving the sum.

a)

int sum = 0;

|  |  |  |
| --- | --- | --- |
| **OPTIONAL** | | |
| **sum** | **index** | **data[index]** |
| 0 | 0 | 1 |
|  | 1 | 2 |
| 5 | 2 | 5 |
|  | 3 | 0 |
| 12 | 4 | 7 |
|  |  |  |
|  |  |  |

int[] data = {1, 2, 5, 0, 7};

for(int index=0; index < data.length; ++index)

{

if (data[index] > 3)

sum = sum + data[index];

}

Sum is: 12

This code has added up the elements in the array that are bigger than 3.

b)

|  |  |  |
| --- | --- | --- |
| **OPTIONAL** | | |
| sum | sourceIndex | index |
| 0 |  |  |
| 1 | 0 | 0 |
|  |  | 1 |
| 2 |  | 2 |
|  |  | 3 |
|  |  | 4 |
|  |  | 5 |
| 3 | 1 | 1 |
|  |  | 2 |
| 4 |  | 3 |
|  |  | 4 |
|  |  | 5 |
| 5 | 2 | 2 |
|  |  | 3 |
|  |  | 4 |
|  |  | 5 |
| 6 | 3 | 3 |
|  |  | 4 |
|  |  | 5 |
| 7 | 4 | 4 |
|  |  | 5 |
|  | 5 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

int sum = 0;

String[] source= {“I”, “think”, “I”, “think”, “know”};

for (int sourceIndex=0; sourceIndex<source.length; ++sourceIndex)

{

// Read the for loop below carefully

for (int index = sourceIndex; index<source.length; ++index)

{

if (source[sourceIndex].equals(source[index])

sum = sum + 1;

}

}

Sum is : 7

This code has added the number of elements in the array to the number of elements in the array that are equivalent.

1. (10 points) Use the algorithm for selection sort that we discussed in class (no other alternatives will be accepted) to show how the int values in the array below are sorted in ascending order (smallest on the left, largest on the right). Show each individual swap on one row.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 3 | 1 | 4 | 6 | 2 | 8 | 9 | 7 | 0 |
| 0 |  |  |  |  |  |  |  |  | 5 |
|  | 1 | 3 |  |  |  |  |  |  |  |
|  |  | 2 |  |  | 3 |  |  |  |  |
|  |  |  | 3 |  | 4 |  |  |  |  |
|  |  |  |  | 4 | 6 |  |  |  |  |
|  |  |  |  |  | 5 |  |  |  | 6 |
|  |  |  |  |  |  | 6 |  |  | 8 |
|  |  |  |  |  |  |  | 7 | 9 |  |
|  |  |  |  |  |  |  |  | 8 | 9 |

1. (50 points; 15 points for a) and b), 20 points for c))

**You may not use ArrayList objects anywhere in this problem.**

Finding an array of words that occur one or more times in a document is the first step towards creating Wordles, like the one below.

[[1]](#footnote-1)

Wordles do not contain common short and simple words (*e.g.* “and”, “this”, “the”, “a”). These words are stored in a file called “ExcludedWords.txt”, written one to a line and kept in sorted order.

Your program should read the excluded words into a perfect sized array. It should then read a file that contains the document to be analyzed. This document contains no punctuation, but may have many words on a single line, and the words may or may not be capitalized. Your program should create an oversized array that contains each unique word that is in the document and not in the excluded array. The array may be, but is not required to be, sorted. The document will contain no more than 10,000 unique words.

For example, if the document contained:

The woods are lovely dark and deep

But I have promises to keep  
And miles to go before I sleep

and miles to go before I sleep[[2]](#footnote-2)

If the array of excluded words contains: {“to”, “i”, “but”, “and”, “or”, “but”} .

The resulting array of size 7 and length (capacity) 10,000 containing: “woods”, “are”, “lovely”, “dark”, “deep”, “have”, “promises”, “keep”, “miles”, “go”, “before”, “sleep” (the order of the words in this array may be different). Notice that words are not repeated in the result list even when they are repeated in the document.

a) Write the method below. The returned array should be perfect sized.

public static String[] readExcludedWords(String fileName) throws FileNotFoundException

{

{

Scanner file = **new** Scanner(**new** File(fileName));

int count = file.nextInt();

file.nextLine(); // gets rid of new line

// Construct the array to the perfect length

String[] result = **new** String[count];

// Now move the data into the array

count = 0;

**while**(file.hasNextLine())

{

result[count] = file.nextLine();

++count;

}

Arrays.*sort*(result); // this isn't necessary if the data

// is already sorted—but it’s not a bad idea

// don’t trust anyone!

file.close();

**return** result;

}

b) Write the method below. The array words is perfect sized. The array excludedWords is oversized. This method can be written in one of two ways. You may use methods from the Arrays class to sort and search the array. Or you may search an unsorted array using a loop.

public static int findWordsInDocument(String fileName, String[] words, String[] excludedWords) throws FileNotFoundException

{

Scanner file = **new** Scanner(**new** File(fileName));

**int** size = 0;

// Process words one at a time

**while** (file.hasNext())

{

String word = file.next();

word = word.toLowerCase();

// The word is in neither place--and therefore needs to be

// added to the words array

**if** (Arrays.*binarySearch*(excludedWords, word)<0

&& Arrays.*binarySearch*(words, 0, size, word) < 0)

{

words[size] = word;

++size;

Arrays.*sort*(words,0, size); // keep it sorted

// so binary search can work

}

// The other option for doing this is to search the array

// using a loop, and then the data doesn't need to be sorted

}

file.close();

**return** size;

}

c) Write the main program by writing the code described by each comment below the comment.

public static void main(String[] args) throws FileNotFoundException

{

// declare and/or construct array for excluded words (3 points)

String[] excludedWords;

// Use the method in a) to read in the excluded words (4 points)

excludedWords = readExcludedWords("ExcludedWords.txt");

// Get the name of the file for the document from the user at the command line (2 points)

Scanner input = new Scanner(System.in);

System.out.println("Enter the file name");

String fileName = input.nextLine();

//Declare and/or construct array for words in document (3 points)

final int SIZE = 10000;

String[] uniqueWords = new String[SIZE];

int uniqueSize = 0;

// Use the method in b) to read the words in the document (4 points)

uniqueSize = findWordsInDocument(fileName, uniqueWords, excludedWords);

// Print the resulting array to the command line (4 points)

System.out.println("The unique words are:");

for (int i=0; i<uniqueSize; ++i)

{

System.out.println(uniqueWords[i]);

}

input.close();

}

1. Source: http://www.web2teachingtools.com/wordle.html [↑](#footnote-ref-1)
2. From Robert Frost’s “Stopping by Woods on a Snowy Evening”, but with alterations to show the program activity better. [↑](#footnote-ref-2)