PEER-LED TEAM LEARNING
COMPUTER SCIENCE

MEETING 10 – STUDENT VERSION
TWO DIMENSIONAL ARRAYS, LINKED LISTS

BARBARA G. RYDER AND PRADIP HARI

Exercise 1. Dating service
Assume you are running a dating service and have identified the following 10 attributes of their prospective date, as being key to securing a 'good match' between individuals:

intelligent
good looking
has a job
good sense of humor
likes to cook
considerate
enjoys being out of doors
prefers dogs to cats
open to new experiences
enthusiastic

We will ask each person using the service to rank these 10 attributes in terms of their importance in a 'perfect date'. The most important attribute will rank 10, the least important will rank 0. It is possible to have a few attributes that an applicant will rank as 'most important'. For our purposes, a rank greater or equal to 7 means 'important' and a rank less than or equal to 2 means 'not important'.

Create a set of six people, to be used as potential matches, by listing their attributes with numbers using the same 10 point scale. For example, Einstein would be a 10 for intelligent, while Tom Cruise might be 10 for good looking. These numbers describe the possible 'date' candidates and will be used to 'match' the requests for dates received from persons using the service. List the six prospective dates as the 1st dimension and the attributes as the 2nd dimension of a two dimensional array assuming we know the indices of each attribute. Also associate another 1 dimensional String array with the same indices, to store the names of the prospective dates.

i. What Java statements are necessary to create these data structures?

ii. Which prospective dates prize a good sense of humor? How would you find the answers to this question? Flowchart your algorithm. Show the key array manipulation steps in your algorithm as Java code.

iii. Choose a person in the group. Have them create a profile by giving their ranks for these attributes. Now find the set of candidates who agree with that person's top three sought after qualities, that is their perfect dates. For example, if Gwendolyn chooses intelligent – 9, considerate – 7, enjoys being out of doors – 8 , then you need to find candidates whose scores for these attributes are greater or equal to the ones Gwendolyn chose. How would you find the answers to these questions? Flowchart your algorithm. Show the key array manipulation steps in your algorithm as Java code.

iv. Choose another person in the group. Have them create a profile. Now find the one candidate who best suits that profile (i.e., whose attributes are closest in value). How would you find the answers to these questions? Flowchart your algorithm. Show the key array manipulation steps in your algorithm as Java code.

Exercise 2: Personal TO-DO Lists
This exercise is going to explore how to build and use lists of **Items** to simulate the functionality of a personal TO-DO list. We will use two classes: **ToDoList** and **Item.** Each **Item** object consists of a String **whatToDo** field which names the activity (i.e., "finish homework assignment") and a integer field **daysUntilDue** that gives the number of days before this **whatToDo** must be completed.

i. Think of the functions you want to perform on your **ToDoList** objects. How will you use your **ToDoList**? Which items will you want to see at the top of the list. Now think about what are the important properties of the Java Lists you use in order to easily implement these functions?

At a minimum, you will need to insert and remove (as they are accomplished) **Items** and **update()** the **daysUntilDue** fields. Give pseudocode for the algorithms for both **insert()** and **remove()** on your lists; work through examples at the board to better understand what both of these methods have to do and what they need to do it.

ii. Since we are keeping track of the days you have left to finish tasks, you will need to perform a **dailyUpdate()** each midnight to update your **daysUntilDue** fields. Flowchart this method. Then show how you would code this method in Java; define any helper methods that you may need. What other methods do you need in your **List** class to implement this method?

iii. What if one of your professors extends the due date on an assignment? What additional functionality do you have to build into your **ToDoList** to accommodate this change? Give pseudocode for the algorithm and then produce Java code for it.

iv. What if you have a **ToDoList** of assignments from school and then you make a **ToDoList** of chores your mother asked you to do by a certain date. How could you merge these two lists producing one master list? Give pseudocode of your algorithm to solve this problem.

Cite this module as: Ryder, B.G., Hari, P. (2012). Peer-Led Team Learning Computer Science: Meeting 10 - Student Version; Two-Dimensional Arrays; Linked Lists. Online at http://www.pltlis.org.
Originally published in *Progressions: The Peer-Led Team Learning Project Newsletter*, Volume 9, Number 1, Fall 2007.