PEER-LED TEAM LEARNING COMPUTER SCIENCE

MEETING 9 – STUDENT VERSION ARRAYS, LOOPS, SEARCHING, SORTING, RUNNING TIME ANALYSIS

BARBARA G. RYDER AND PRADIP HARI

Exercise 1: Poker game simulation

Recall from class that you already have seen Card objects with face and suit fields defined using enumerated types. For this exercise, we will use a Card object that has a Suit enumerated type, but an integer value for the face, so that we can check for consecutive face values.

To do this simulation we will need a **Deck** class which consists of an array of Card objects. In order to play a game of poker, a Deck object will have to be initialized by randomly filling it with Card objects; periodically during the game the Deck will be shuffled. Therefore, the methods in the Deck class are**: (i) a constructor**, which we assume initializes the Deck by iterating over all suits and all faces in a suit, (ii) **shuffle()**, which simulates a player rearranging the Deck by shuffling the Cards, so they are in no particular order, and (iii) **drawCard()**, which removes cards from the top of the Deck when they are picked by a player.

a. Think about how to implement **shuffle()** using arrays. There are ways to do this either with two arrays or one array.   Flowchart your ideas. Try writing the code for your shuffle() method. Use **PokerDriver** (a program we are providing to you) to test it out.

In poker we will consider identifying hands that contain *pair(s)* (i.e., 2 of the same-valued card in different suits) or contain a *flush* (i.e., all cards in the same suit), or contain a *straight* (i.e., all cards have consecutive face values). We can represent a hand by an array of **n** Card objects, where **n** is usually 5 but may be 7 or some other value.

Parts b-e ask you to figure out an algorithm to identify each type of hand and then flowchart the algorithm. First, think about how to solve the problem. Second, when you figure out a solution, then consider its efficiency. Can you arrange the cards to make your algorithm work better?

b. Think about an algorithm to determine whether a hand contains a *flush*.

c. Think about an algorithm to determine whether a hand contains a *straight*.

d. Think about an algorithm to determine whether a hand contains a single *pair*. This is trickier than it seems at first, because to contain a *pair,* a hand should not contain three- or four-of-a-kind.

e. Think about an algorithm to determine whether a hand contains two different *pairs*.

Exercise 2: Time analysis of everyday tasks
Think of working as a library page who has to shelve returned fiction books in the library.

a. Describe an algorithm that tells how you would do this task. As a group, name 10 novels (title and author) and hand-simulate the algorithm on these books.

b. We want to calculate the cost (in effort) of this task. Assume that you start shelving at the A authors and continue through to the Z authors. Assume there is 1 bookcase of books for each letter of the alphabet containing all authors whose last name starts with that letter. Assume it costs 5 steps per letter of the alphabet to walk past each bookcase on the way to the first letter of the author of the book you are to shelve. For example, it costs 10 steps to get to bookcase C starting at bookcase A, in order to shelve a book by James F. Cooper. Assume that you return to bookcase A after shelving each book. With these assumptions, , calculate the costs of shelving your selected 10 novels,

1. if they are not in any order on the book cart?
2. if they are put into sorted order by author's last name on the book cart?

Given your estimates for 1, 2 above, then can you estimate the *worst case* cost of shelving **n** books

- if they are not in any order on the book cart?
- if they are put into sorted order by author's last name on the book cart?

Describe the pattern of author last names for the books to be shelved associated with this *worst case* cost.

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