CSE 2010, HW1, Spring 2025
Due Thu Jan 30 at the start of your lab section; Canvas: HW1
Due Thu Jan 30 at the end of your lab section; Canvas: HW1a

An online retailer has a warehouse, where workers pick items from storage shelves to fulfill customer orders. The picked items can later be packed and shipped to the customers. Each order could have a different number of items and different categories of items. To save time in making fewer trips, a worker can handle more than one order. How would you design a system to assign workers to customer orders?

For this assignment, for simplicity:

- Assume 1 packing station, 2 categories of items (books and electronics), 5 initially available workers in this order: Alice, Bob, Carol, David, Emily, the capacity of each worker on one trip is 10 items, and each order has at most 10 items.
- A worker can be assigned to more than one customer order ("bundling" the orders) if the next order has the same category(ies), but the total number of items does not exceed 10. For example, if the items are all books and no more than 10 books after bundling, they may be bundled. The system should decide to bundle or not when an order is received (ie, at a particular time, future orders are not known).
- If a worker has the capacity to bundle, he/she waits to potentially bundle the next order, but should not wait for more than 5 minutes.
- Only succcessive orders may be bundled; ie. do not skip an older order and bundle with a newer order. More than 2 successive orders may be bundled.
- Assume each worker needs 1 minute to find and pick each item in the same category, and an additional 5 minutes of traveling time between two categories or between the packing station and a category.

To separately manage the **customer orders**, **available workers**, and **worker assignments**, use 3 singly linked lists. We will evaluate your submissions on code01.fit.edu so we strongly recommend you to test your programs on code01.fit.edu. To preserve invisible characters, we strongly recommend you to download, NOT copy and paste, input data files.

Input: To simulate the customer orders, an input file contains the customer orders in the same directory as your program file called HW1.java that has the main method. Your submission takes the input file name as a command-line argument. Each line is one of the following:

- CustomerOrder $orderTime\ customer\ numberOfBooks\ numberOfElectronics$
- PrintAvailableWorkerList $printTime$
- PrintWorkerAssignmentList $printTime$
- PrintMaxFulfillmentTime $printTime$

For simplicity, time is in HHMM format (HH: 00-23 and MM: 00-59), the leading zero is optional. Fulfillment Time is the amount of time between when an order is made/received at a packing station and when a worker finishes picking items for the order(s) and walking back to the packing station. MaxFulfillmentTime is the longest Fulfillment Time among all the orders so far. AvailableWorkerList is in availability order. Ordering of WorkerAssignmentList is the expected completion time of a worker.

Output: The program prints events to the standard output (screen). Each event is on one line and possible events are:

- CustomerOrder $orderTime\ customer\ numOfBooks\ numOfElectronics$
- WorkerAssignment $assignmentTime\ worker\ customer$
- OrderCompletion $orderCompletionTime\ customer$
- AvailableWorkerList $time\ worker1\ worker2$ ...
- WorkerAssignmentList $time\ worker1$:$customer1\ worker2$:$customer2a$,$customer2b$ ...
- MaxFullfillmentTime $time\ numberOfMinutes$

Submission: Submit HW1.java that has the main method, (modified or your own) SinglyLinkedList.java and other program files. Submissions from individual students are due at the beginning of their respective lab sections via assignment HW1 (see the top).

During the lab session on the due date, we encourage students to bring test cases (beyond the sample input) to test and improve each other's program in the group. Improved programs are submitted via assignment HW1a, which is due at the end of the lab section (see the top). Your program is mainly evaluated based on HW1. Improvement on test cases will receive half credit. Specifically, $testCaseImprovement(hw1) = testCaseScore(hw1a) - testCaseScore(hw1)$; $testCaseScore(hw1) = testCaseScore(hw1) + testCaseImprovement(hw1)/2$.

Note the late penalty on the syllabus if you submit after the due date and time as specified at the top.