Computer Science 1 – Program 4
UCF-Mart (Stacks & Queues)
Assigned: 10/13/10
Due: 10/27/10 (Wednesday) at 11:55pm (WebCourses time)

The Problem

UCF lacks a credible grocery store on campus. After a year of getting lousy over-priced food in the convenience store in the Student Union, you and all of your friends are frustrated. One of your friends majoring in Business comes up with the brilliant idea of opening a grocery store on campus. But, they need your help to decide if their idea is viable or not. Your job will be to run various simulations of customers in line.

The model for the store is as follows:

There will be three lines in operation or customers to buy their groceries. Each line will have a capacity of 8 customers. The first line (line A) will only be for customers with 10 or fewer items. The other two lines (line B and C) will be for all customers. When a customer is ready to check out, here is how they decide what line to go to:

If the customer has 10 or fewer items and the "Express" line has space, they enter that line.

In all other cases, customers will alternate getting into line B and line C (starting with line B), unless one of the lines is full, in which case they will enter the line that is not full. It will be guaranteed that no more lines will be needed to serve all the customers.

For each customer, you'll be given the following information:

1)His/her name (an uppercase alphabetic string of fewer than 30 characters)
2)The number of items they are buying
3)The time they arrive in line, in seconds after 1 pm. (This makes reading in the data easier.)

For the purposes of the simulation, the store will be open from 1pm to 8pm and all customers who get in line by 8pm will get to buy their items (even though some of them might technically check out with the cashier after 8pm.) It is guaranteed that none of these late checkouts will go past 11:59:59pm. (Thus, all customers will get into a line in between 1pm and 8pm inclusive, and all customers will get checked out in between 1pm and 11:59:59pm, inclusive.)

The information in the input file will be given in the order that the customers arrive to get into a line. At that time, they should be placed in the appropriate line immediately. No two customers will arrive to get into line at the exact same time or have the same name.

The number of seconds it takes a customer to check out of line once they've gotten to the front of it is 40 plus six times the number of items they are buying. Thus, if they are buying 9 items, it will take them $40 + 6(9) = 94$ seconds.

Your task will be to print out an output log of each action for each of the lines in UCF-Mart, namely a list of when each customer gets in line and when each one checks out.

Input File Specification (input.txt)

You will read in input from a file, "input.txt". The name MUST BE "input.txt". Have this AUTOMATED. Do not ask the user to enter "input.txt". You should read in this automatically. (This will expedite the grading process.)

The input file has a single positive integer, *n*, on its first line, specifying the number of total days to run the simulation. Each of the simulations will follow.

The first line of each simulation will contain a single positive integer, *k*, representing the number of customers who get into line for that day of the simulation. The following *k* lines will contain information about each customer and will be in the order in which they arrive to get in line. The first piece of information on each of these lines will be a non-negative integer (less than or equal to 25200) representing the number of seconds after 1pm the customer arrives to get into line. This is followed by the name of the customer (all uppercase letters with a length of fewer than 30 characters). The final piece of information on each line will be the number of items they are buying, which will be a positive integer less than 1000. As previously mentioned, the data will be such that the last checkout occurs at or before 11:59:59pm. Also, the data will be set up so that no two actions (customers getting into or out of line) occur at exactly the same time. All three pieces of information will be separated by a space on each line.

Output Specification

*****NOTE***: You should generate your output to a FILE that you will call "out.txt".**

For each input case, print out a header with the following format:

```
Day #d:
```

where *d* represents the day of the simulation ($1 \leq d \leq n$).

Follow this with a blank line.

The following *2k* lines will give information about a customer either getting into a line, or checking out of a line. These lines should be printed in the order in which the actions occur.

If a customer is checking in, print out a statement with the following format:

```
TIME:  Customer NAME checking in line X.
```

where TIME is the time the customer checked in, NAME is the name of the customer, and X is the letter of the line (A, B, or C) that the customer is entering.

The time should be printed out in the following format:

```
(H)H:MM:SSpm
```

The first one (or possibly two) digits represent the hour. This is followed by a colon and then the next two digits represent the minute (e.g. `printf("%.2d", 3);`). Finally, the last two digits represent the second, printed in a similar format to the minute. (It probably makes sense to have a function that takes in as input the number of seconds after 1pm and in turn prints out the corresponding time in this format.)

If a customer is checking out, print out a line with the following format:

```
TIME:  Customer NAME checking out of line X waited Y seconds.
```

TIME, NAME and X are defined previously, and Y is a positive integer representing the number of seconds from when the customer got in line to when they *finished* checking out. (Thus, this wait time *includes* the time it took for them to get their groceries scanned and paid for. So, the fastest possible wait time is 46 seconds, based on the formula given.)

Follow each day's output with two blank lines.

*****WARNING*****

Your program MUST adhere to this EXACT format (spacing capitalization, use of dollar signs, periods, punctuation, etc). The graders will use very large input files, resulting in very large output files. As such, the graders will use text comparison programs to compare your output to the correct output. If, for example, you have two spaces between in the output when there should be only one space, this will show up as an error even through you may have the program correct. You WILL get points off if this is the case, which is why this is being explained in detail. Minimum deduction will be 10% of the grade, as the graders will be forced to go to text editing of your program in order to give you an accurate grade.

Implementation Restrictions

Name the file you create and turn in *UCFmart.c*. You must create a struct that stores a queue. You must write functions that operate on this struct analogous to the queue functions shown in class. Your program must make a queue variable for each of the lines in the simulation. You may implement your queue with either an array or a linked list, but make sure you find a reasonable solution to limiting the line sizes to 8.

Sample Input

```
2
2
5 ANNA 12
2000 BOB 5
3
10 ANNA 16
500 BOB 32
2000 CAROL 9
```

Sample Output
PAY ATTENTION TO THE EXACT FORMAT (SPACING, PUNCUATION, ETC.)

```
Day #1:

1:00:05pm:  Customer ANNA checking in line B.
1:01:57pm:  Customer ANNA checking out of line B waited 112 seconds.
1:33:20pm:  Customer BOB checking in line A.
1:34:30pm:  Customer BOB checking out of line A waited 70 seconds.


Day #2:

1:00:10pm:  Customer ANNA checking in line B.
1:02:26pm:  Customer ANNA checking out of line B waited 136 seconds.
1:08:20pm:  Customer BOB checking in line C.
1:12:12pm:  Customer BOB checking out of line C waited 232 seconds.
1:33:20pm:  Customer CAROL checking in line A.
1:34:54pm:  Customer CAROL checking out of line A waited 94 seconds.
```

Grading Details

Your program will be graded upon the following criteria:

1) Correctness.
2) Your use of functions. Even if your output is correct, you won't receive full credit if your program doesn't make appropriate use of functions.
3) Use of queues. Since the purpose of this assignment is to teach you to use queues, you will not receive credit if you don't use them.
4) Your programming style, comments, and use of white space. (Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor you could get 10% or 15% deducted from your grade. We won't be stupendously picky on this point, but your code should look reasonable.)

Deliverables

Turn in a single file, *UCFmart.c*, over WebCourses that solves the specified problem. Make sure to include ample comments and use good programming style, on top of the requirements that are given in the program description above.