CS-212: LAB 7 – Queues

Learning Objective:

- Running a simulation with queues to generate an input for this lab
- Using the input file from the simulation to produce two output files
- Bit manipulation

This lab has five parts:

- Part-A: Generating your input file
- Part-B: Creating your header file
- Part-C: Getting started with your main function
- Part-D: Writing the main
- Part-E: Writing the functions

Lab 7 Setup:

1. Open a new terminal window to SSH into your account on the [bingweb.binghamton.com] server. When logged in, make sure you are in your home directory by typing the following command if you are not already there. The "~" is just a shortcut symbol that means home directory:
   cd ~

2. Change your current directory to the "CS212" directory by typing:
   cd CS212

3. Confirm that you are in the "CS212" directory by typing:
   pwd

4. Create a directory for Lab7.
   mkdir Lab7

5. Move into the new directory
   cd Lab7
6. Create a new c file for you to write your lab.
   > <Last Name><FirstInitial>_Lab7.c
   > <Last Name><FirstInitial>_Lab7_Functions.c
   > <Last Name><FirstInitial>_Lab7.h

PART A: GENERATING YOUR INPUT FILE

Before you start this lab, you will need to generate your input file from the simulation program that is available to you on Brightspace.

Run the program and make sure your data meets the following requirements:

- You must have at least 100 happy customers
- You must have a minimum of 20 angry customers
- The number of angry customers must be less than half of all your customers

To achieve these requirements, you may have to alter the inputs found in "Inputs.h".

It is recommended that your read the IPO found in "Simuation.c" before attempting to alter the inputs.

When the output from the simulation meets the requirements, copy the output file, "SimulationOutput.txt", to your lab directory and rename it to "Lab7_Input.txt".

PART B: CREATING YOUR HEADER FILE

Inside your header file you should include:

- Safeguards
- Include statements
- Constants / Macros
- Structure definitions (*see below for additional information*)
- Prototypes

Setting up your structures:

HINT: You should review the playbook notes on queues and "*Sample Code For Queues.c"*

Create a structure recordType and include the following information:

- ID Integer
- reason code Integer
- message char array of size 50 (use a constant/macro for the size)

Create your node and include the following information:

- An instance of your recordType
- Node pointer

Create a structure named queue and include the following information:

- Node pointer for the front of the queue
- Node pointer for the back of the queue

PART C: Getting started with your main function

At the top of your main, create the following variables (You may need others):

- Queue for happy customers
- Queue for angry customers
- Node pointer for a new record
- File pointer for an input file
- File pointer for an output file for the happy customers
- File pointer for the output file for the angry customers

Some general thoughts to consider before coding the lab:

- The first part of your program will process the input file. You will want to use a file processing loops (since you don't know how many customers are in the file).

- You will read in one integer at time. Each integer is a code with all the information you need embedded inside it. (*Details on how to decode the integer are below*)

- You will need two queues, one for angry customers and one for happy customers. Queues MUST use dynamic memory. You may not use arrays.

- After you have all nodes in the two queues, dequeue the queues writing the information to the output file. (Hint: Since the queues should have different number of nodes in them, process one queue at a time)

PART-D: WRITING THE MAIN

When writing the main, think about it in 3 parts.

- Part 1 – The input file
- Part 2 – Process the angry customers
- Part 3 – Process the happy customers

The input file:

Check your file pointer for NULL and start a file processing loop to process the input file

- Read in the first integer
- Create a node and decode the integer.
- Insert the node into the appropriate queues

Process the angry and happy customers

NOTE: You should set up your functions so that you can call them for both angry and happy customers.

Print the heading to the output file for the angry/happy customers
Print the column heading to the output file for the angry/happy customers

Loop until angry/happy customers queue is empty
- Dequeue one node
- Print the data to the output file
- free the node

Close your files

PART-E: WRITING THE FUNCTIONS

NOTE: It is up to you to determine the parameters for most of the functions. There are few functions where describe the parameters for you.

FUNCTIONS: PrintHeader, CenterString, PrintDivder

These functions can be copied into your lab. All functions must be capable of sending their output to an output file or the screen.

FUNCTION: PrintTableHeader

This function will print the header to your output table. This function should be called twice, once with the angry customer output file pointer and once with the happy customer output file pointer.

Sample output from this function:

```
Customer ID      Reason
--------------------------------------------------------------------------------
```

FUNCTION: PrintRowOfData

This function will print a customer ID and the message set based on the reason code.
This function will be called twice, once for the angry customers and once for the happy customers.
(The function to set the message will be described below)

FUNCTION SetAngryMessage
Parameter: nodeType pointer

Use as switch statement to copy (use the strcpy function) to copy one of the following messages into the message member of your structure.

CODE	Message
0	This store is terrible, I'm never coming back!
1	I'm too busy for this! Maybe I come back later.
2	Too much to do, too little time!
3	<yawn> I'm too lazy to wait in line.

HINT: Put a new line character at the end of each string

FUNCTION: SetHappyMessage
Parameter: nodeType pointer

Use as switch statement to copy (use the strcpy function) to copy one of the following messages into the message member of your structure.

CODE	Message
0	AWESOME SERVICE! Can't wait to come back.
1	Checkout was a breeze!
2	Got what I needed, very good store!
3	Smiling customer will be happy to return

HINT: Put a new line character at the end of each string

FUNCTION: Dequeue
Parameter: queueType pointer to a line (Maybe the happy customers or the angry customers)

This function will remove one node from the front of the queue

FUNCTION: EnqueueNewNode
Parameters: queueType * for happy customers, queuetype * for angry customers, nodeType pointer for a node, unsigned integer for the code

Check the code to see if the customer is either angry or happy

If the customer is angry, call Enqueue passing in the angry customer queue

If the customer is happy, call Enqueue passing in the happy customer queue

FUNCTION: Enqueue
Parameters: queueType pointer for a list of customers, nodeType pointer that points to a new node

This function will enqueue one new node.
The function will be called by EnqueueNewNode.

FUNCTION: CreateNodeNode
Parameters: nodeType pointer and an unsigned integer that will represent the code

This function will decode the code and set the data in the structure pointed to by the pointer.

DECODING THE INTEGER CODE

The code is made up of 32 bits. The first 16 bits are displayed below.

BIT:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

- If bit 7 of the code is equal to one, then you have an angry customer
  Set the reason code in your new node to the integer represented by bits 1 and 2.

- If bit 6 of the code is equal to one, then you have a happy customer
  Set the reason code in your new node to the integer represented by bits 3 and 4.

- The customer ID is in bits 8-15

- If you have an angry customer, call SetAngryMesage

- If you have a happy customer, call SetHappyMessage

SAMPLE OUTPUT

The following is sample output from an angry customer queue

```
*********************************************************************************************************
                                    Binghamton University
                                Program written by: Prof. Foos
                                          Lab 7
*********************************************************************************************************

Customer ID      Reason
---------------------------------------------------------------------------
    22      <yawn> I'm too lazy to wait in line
    23      <yawn> I'm too lazy to wait in line
    25      This store is terrible, I'm never coming back!
    26      This store is terrible, I'm never coming back!
    28      Too much to do, to little time!
    29      I'm too busy for this! Maybe I come back later.
```

NOTE: Remember the input data is random data, so your output will be different.

RUNNING YOUR PROGRAM

Your program must run in gcc on the unix operating system with 0 errors and 0 warnings.

To run your program:

```
gcc *.c -Wall
```

This will create an output file named a.out

Valgrind must run with 0 errors and 0 warnings.

To run Valgrind:

```
valgrind --leak-check=full ./a.out
```

FILES TO SUBMIT TO BRIGHTSPACE

- <Last Name><First Initial>_Lab7.c
- <Last Name><First Initial>_Lab7_Functions.c
- <Last Name><First Initial>_Lab7.h
- HappyCustomers.txt
- AngryCustomers.txt

ADDITIONAL PARTIAL CREDIT GRADING RUBRIC:

NOTE: The full rubric is on Brightspace under Misc

None.

Note to Grader: Any partial credit outside of the rubric must be approved by the teacher.