CptS 122 – Data Structures

Lab 4: Data Structures, Dynamic Linked Queues, and Unit Tests in C

Assigned: Tuesday, June 20, 2017
Due: At the end of the lab session

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Design, implement and test a dynamic queue implemented with a singly linked list in C
- Compare and contrast dynamic linked lists, dynamic linked stacks, and dynamic linked queues
- Summarize the advantages of applying a queue within certain applications
- Describe the operations applied to a queue including
    1. enqueue ( )
    2. dequeue ( )

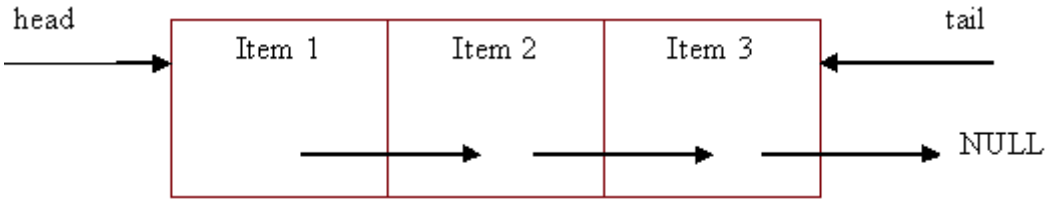
II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- Analyze a basic set of requirements for a problem
- Compose a small C language program
- Create test cases for a program
- Apply and implement structs in C
- Apply and implement pointers in C
- Apply and implement dynamic memory in C
- Design and implement a dynamic singly linked list
- Design and implement a dynamic stack implemented with a singly linked list

III. Overview & Requirements:

This lab, along with your TA, will help you navigate through designing, implementing, and testing a dynamic queue implemented with a linked list. **Recall, a queue data structure is a restricted linked list, where only the front or head node in the queue may be processed and then removed, at any given time. However, only nodes may be added to the end, back, or tail of the queue. A queue is referred to as a first-in, first-out (FIFO) structure as a result of this constraint.** Furthermore, the operations of a queue must adhere to this restriction. An `enqueue()` operation adds a node to the tail of the queue and a `dequeue()` operation removes a node from the head of the queue. We will visualize a queue in the following way:



Labs are held in a "closed" environment such that you may ask your TA questions. Please use your TAs knowledge to your advantage. You are required to move at the pace set forth by your TA. Please help other students in need when you are finished with a task. You may work in pairs if you wish. However, I encourage you to compose your own solution to each problem. Have a great time! Labs are a vital part to your education in CptS 122 so work diligently.

Tasks:

1. For the following problem define a struct `queueNode` with data of type `char *`. Also define a `struct queue` with two pointers: one for the *head* and one for the *tail*. Both pointers are `struct queueNode *` types. Implement the following operations for your queue data structure:

    1. `isEmpty()` - a predicate function which checks to see if the queue is empty; returns true if the queue is empty; false otherwise

    2. `enqueue()` - inserts a node to the tail of the queue; the node is allocated dynamically; returns true if the memory was allocated for a node, false otherwise

    3. `dequeue()` - deletes a node from the head of the queue; returns the data in the node; precondition: queue is not empty (isEmpty () must be called before dequeue () is called)

    4. `printQueueRecursive()` - recursively prints out the data in the queue

2. Test your application. In the same project, create one more header file `testQueue.h` and source file `testQueue.c` (for a total of at least five files). The `testQueue.h` file should contain function prototypes for test functions you will use on your queue functions. The `testQueue.c` source file should contain the implementations for these test functions. You will have at least one test function per application function. For example, you will have an application function called enqueue`()` (or a function very similar) that is used to insert a node into the tail of the queue. In this task, you will need to create a test function called `testEnqueue()` that passes in various data directly into `enqueue()` to see if it works as intended. You will also want to test these functions on empty and non-empty queues.

3. Work on the current programming assignment.

IV. Submitting Labs:

- You are not required to submit your lab solutions. However, you should keep them in a folder that you may continue to access throughout the semester. You should not store your solutions to the local C: drive on the EME 120/128 machines. These files are erased on a daily basis.

V. Grading Guidelines:

- This lab is worth 10 points. Your lab grade is assigned based on completeness and effort. To receive full credit for the lab you must show up on time and continue to work on the problems until the TA has dismissed you.