**Lab 6 - Linked Lists, Stacks, and Queues**

Part 1: Model Train (from in-class exercise)

# Part A: Create a program representing a model train

# Create structs to model a train. You will need:

* + - A train struct to represent the whole train that contains a pointer to the first train car;
    - A car struct for each train car with: (1) cargo weight (this can be an int) and (2) a pointer to the next car so you can access each car;
    - A way to know when you are at the last car.
  + Using scanf to let user input the number of cars in your train and the cargo weight of each car (Input number may be 0 or 1).
  + You need to use malloc to allocate a separate heap memory space for each car. (You cannot use malloc to allocate a large array or use stack memory for cars).
  + Print the memory address and cargo weight of each car, then free the memory for your train.

Part 2: Linked Lists

# Part A: Singly Linked List

* + Create a singly linked list using list and node structs. You must create your linked list on the heap (using malloc). Your linked list should have the following operations:
    - createNode - initializes a node struct
    - createList - initializes the linked list struct with the following attributes:
      * A pointer to a head node
      * returns a pointer to a LinkedList struct created on the heap
    - Front - returns a pointer to the first node of the list
      * node \* front(List \* list);
    - Insert - inserts an element at a specified index in the list and returns a pointer to that index if successful. If the index is out of bounds, you may handle it however you wish, but your program should not crash
      * node \* insert(List \* list, int index, Data value);
    - Delete - deletes an element from a specified index in the list.
      * int remove(List \* list, int index);
  + Generate 10 random numbers and store them in the linked list. After each number, you should print out all the current values in the linked list.

# Part B: Doubly Linked List

* + Alter your linked list so it is a doubly linked list.
    - A doubly linked list only adds features to the singly linked list.
  + Add the following functionality
    - searchForward - search for the value starting from the front node
    - searchBackward - search for the value starting from the back node.
  + Generate 10 random numbers and store them in the linked list. Prompt the user to enter a search value, and print the number of steps before the value is found both forward and backward.
    - Do not worry about duplicate values in the list. In this case the number of steps won't equal the size of the list.
    - If the value is not found, print, “The value was not found”
  + Upon exiting, use your delete function to clean up the linked list memory

Part 3: Stacks and Queues

# Part A: Stack

* + Create a Stack struct that is a wrapper for your linked list
  + You should implement the following functions that take a stack:
    - void push(Stack \* stack, int value)
    - int pop(Stack \* stack)
  + Prompt the user to input 5 integers, and use the stack to reverse the integers
  + Print the result to the screen.

# Part B: Queue

* + Create a Queue struct that is a wrapper for your linked list
  + You should implement the following functions that take a stack:
    - void enqueue(Queue \* queue, int value)
    - int dequeue(Queue \* queue)
  + Prompt the user to enter 5 integers, and store it in the queue.
  + Use your dequeue function to print all 5 integers to the screen.

Part 4 - Submission

Create a tar archive with the command “tar -czvf lab6.tar.gz .”, and then email your archive to bu580u2017@gmail.com and cc your TA dmu1@binghamton.edu before the submission deadline. Make sure you do not include the executable in your archive (make clean before creating the archive). Late assignments will not be accepted under any circumstances. Plan to turn in your assignments early.

Demo your lab before the demo deadline (after the submission deadline) by downloading your submission from class Gmail and extracting your archive with the command “tar -xvf lab6.tar.gz”. Then compile (with your makefile), and run your code, show your source to your TA, and answer any questions your TA may have.

Grading Guidelines

## Part 1:

* + Part A: 3 points
* **Part 2:**
  + Part A: 2 points
  + Part B: 2 points
* **Part 3:**
  + Part A: 1 point
  + Part B: 1 point

## Style Guidelines and Memory Leaks

* + Follows Style Guidelines: 1 point
  + Valgrind Shows Memory Leak: -2 points

Submission Deadline: 11:59pm 11/28/2017 EST

Demo Deadline: 2:00pm 12/8/2017 EST