Oregon State UNIVERSITY | College of Engineering

CS261 Main/Programming Assignment 3

This assignment is comprised of 3 parts:

Part 1: Linked List Deque and Bag implementation

First, complete the linked list implementation of the deque (as in Worksheet 19) and bag ADTs (Worksheet 22 due in week 4). To do this, implement all functions with the `// FIXME...` comments in `linkedList.c` .

Grading (30 pts):

- `init` -- 2 pts
- `addLinkBefore` -- 3 pts
- `removeLink` -- 3 pts
- `linkedListAddFront` -- 2 pts
- `linkedListAddBack` -- 2 pts
- `linkedListFront` -- 2 pts
- `linkedListBack` -- 2 pts
- `linkedListRemoveFront` -- 2 pts
- `linkedListRemoveBack` -- 2 pts
- `linkedListIsEmpty` -- 2 pts
- `linkedListPrint` -- 2 pts
- `linkedListContains` -- 3 pts
- `linkedListRemove` -- 3 pts

Part 2: Circularly Linked List Deque implementation

In this part, you will implement the Deque ADT with a Circularly-Doubly-Linked List with a Sentinel. As you know, the sentinel is a special link, does not contain a meaningful value, and should not be removed. Using a sentinel makes some linked list operations easier and cleaner in implementation. This list is circular, meaning the end points back to the beginning, thus one sentinel suffices. Implement all functions with the `// FIXME...` comments in `circularList.c`.

Grading (30 pts):

- `init` -- 1 pts
- `createLink` -- 1 pts
- `addLinkAfter` -- 2 pts
- `removeLink` -- 2 pts
- `circularListAddBack` -- 2 pts
- `circularListAddFront` -- 2 pts
- `circularListFront` -- 2 pts
- `circularListBack` -- 2 pts
- `circularListRemoveFront` -- 2 pts
- `circularListRemoveBack` -- 2 pts
- `circularListDestroy` -- 2 pts
- `circularListIsEmpty` -- 2 pts
- `circularListPrint` -- 2 pts
- `circularListReverse` -- 6 pts (Importantly, you must perform the list reversal in place and may not allocate any new memory in this function)

Part 3: Linked List Stack implementation using two Queues

This is the most challenging part of this assignment. Please note that you will not be provided any skleton code/ header file to solve this problem. In this part, you will use two instances of Queue data structures to implement a Stack. Also, you will use only one C file (stack_from_queue.c) to design the entire interface.

Make sure that you implement all the stack functions provided in Worksheet 17 using the queue functions provided in Worksheet 18. You should also be able to test stack_from_queue implementation by pushing, checking the top, and popping. Finally, you (so are we) should be able to run the file using a single gcc command -

```
gcc -g -Wall -std=c99 -o  stack_from_queue  stack_from_queue.c
```

// All by yourself `stack_from_queue.c`

Grading(40 pts):

Implementation provided in `stack_from_queue.c`

Submission

As usual do not make any modifications to the header files or include any additional headers, and make sure everything compiles and runs on flip. Submit the following files:

- `linkedList.c` -- your linked list deque and bag implementation.
- `circularList.c` -- your circularly linked list deque implementation.
- `stack_from_queue.c` -- your linked list stack implementation.