CSE 1320: Intermediate Programming
University of Texas at Arlington
Spring 2021
Dr. Alex Dillhoff

Assignment 7

This assignment focuses on enums, stacks, and queues.

Your code must compile without any warnings or errors and run without segmentation faults to receive credit. Additionally, any allocated memory must be freed. Points will be taken for inconsistent formatting.

1. A repair company has observed that the number of repeat repairs have increased. To reduce the number of mistakes when fixing a product, the company has contracted you to come up with a program that can trace the disassembly steps of a product so that the re-assembly can be output.

   To do this, you will write a program that uses a stack. Each disassembly step is pushed onto the stack. When it is time to re-assemble the product, the stack can be popped until no items are remaining. There may be some disassembly steps that do not have a one-to-one mapping with an assembly step. For these cases, use an enumerated type to indicate if the step is optional or required.

   Data Format

   ```
   enum priority {
       OPTIONAL,
       REQUIRED
   };

   typedef struct {
       char *desc;
       enum priority p;
   } instruction;
   ```

   Other Requirements

   - When printing the assembly, print only steps that are **required**.
   - Each instruction should be a node in a linked list-based stack.
   - Nodes should be managed using dynamic memory allocation.

   Example Run

   ```
   # Add step
   1. Add step
   2. Undo step
   3. Assemble
   ```

```
4. Quit
> 1
Description: clean surface
Priority (0, 1): 0

# Undo step
1. Add step
2. Undo step
3. Assemble
4. Quit
> 2
Step "take front cover off" removed.

# Print disassembly
1. Add step
2. Undo step
3. Assemble
4. Quit
> 3
1) remove battery
2) take front cover off
```

2. Create a quest log that stores an adventurer's quests that have yet to be completed. The log should be implemented as a queue. A menu should be presented to the user with the options to add a new quest (enqueue), view the current quest, and mark the current quest completed (dequeue).

   Other Requirements

   - Only the quest at front of the queue can be viewed.
   - After viewing the quest, the user can either choose to mark it as complete or go back to the menu.
   - The queue can be implemented using either a dynamic array or linked list.

   Example Run

```
1. Add Quest
2. View Next
3. Quit
> 2
Find Mankrik's Wife
1. Mark as Complete
2. Go back
> 2
1. Add Quest
2. View Next
3. Quit
> 3
```

Create a `zip` file using the name template `LASTNAME_ID_A7.zip` which includes the all required code files. Submit the `zip` file through Canvas.