Practice 3. Queues

[CSE2010] Data Structures

Department of Data Science

Overview

- Implement a **queue** by using (1) an array or (2) a linked list.
- Functions
  - Enqueue a given integer to the queue: O(1) time
  - Dequeue the integer from the front of the queue, and retrieve that integer: O(1) time
  - Retrieve the integer at the front of the queue: O(1) time

- Implement a **stack** by using queues you implemented: StackViaQueues
- Functions
  - Push a given integer to the stack
  - Pop the integer at the top of the stack, and retrieve that integer
  - Retrieve the integer at the top of the stack
- What is the running time of each operation of this stack?

Input of Queue

- Each line represents a single operation.
  - **E<space>[int]**
    - If enqueue fails, immediately terminate the program with the error message.
    - Otherwise, write to the output file every element (from the front to the end) in the queue after this operation.
  - **D**
    - If dequeue fails, immediately terminate the program with the error message.
    - Otherwise, write to the output file every element (from the front to the end) in the queue after this operation.
  - **F**
    - If retrieving the front fails, immediately terminate the program with the error message.
    - Otherwise, write the top element to the output file.

Input of StackViaQueues

- Each line represents a single operation.
  - **U<space>[int]**
    - If push fails, immediately terminate the program with the error message.
    - Otherwise, write to the output file every element (from the bottom to the top) in the stack after the push operation.
  - **O**
    - If pop fails, immediately terminate the program with the error message.
    - Otherwise, write to the output file every element (from the bottom to the top) in the stack after the pop operation.
  - **T**
    - If top fails, immediately terminate the program with the error message.
    - Otherwise, write the top element to the output file.

Input and Output

- Each line represents to the result of the corresponding line of the input file.
- Input File & Output File

```
E 9
E 4
E 2
E 8
F
E 7
F
D
F
D
F
```

```
9
9 4
9 4 2
9 4 2 8
9
9 4 2 8 7
9
4 2 8 7
4
2 8 7
2
```

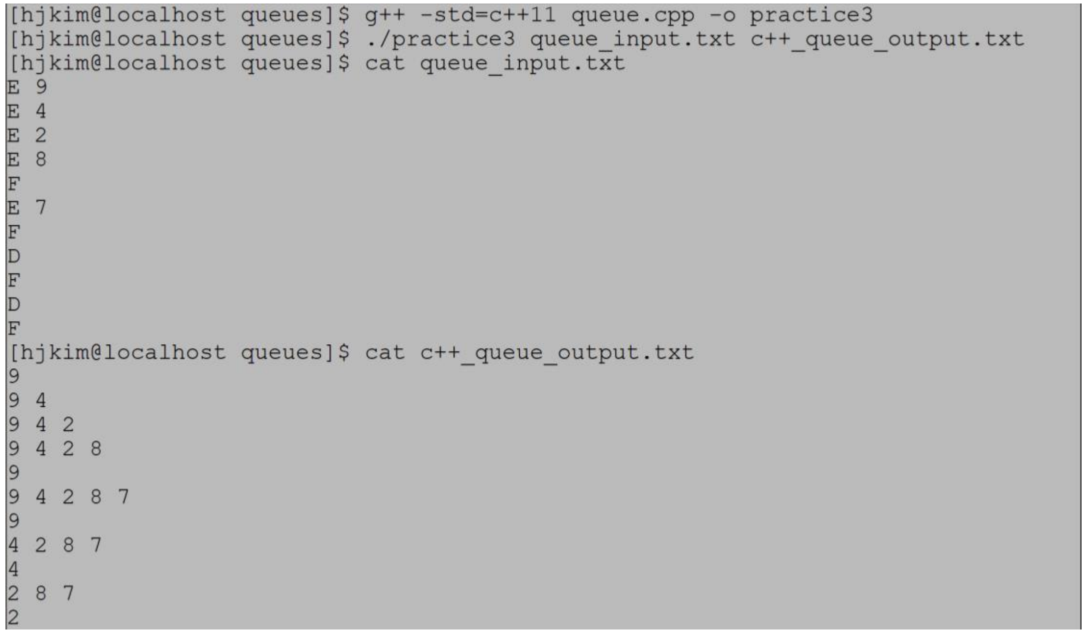
- Input File & Output File

```
U 9
U 4
U 2
U 8
T
U 7
T
O
T
O
T
```

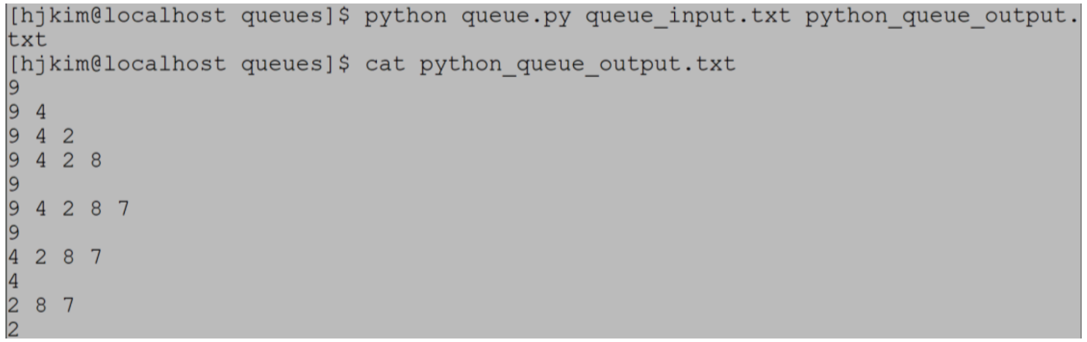
```
9
9 4
9 4 2
9 4 2 8
8
9 4 2 8 7
7
9 4 2 8
8
9 4 2
2
```

Running Example

- Please refer to the last practice on file I/O to implement this exercise.
- You can implement your code based on the skeleton code provided.



```
[hjkim@localhost queues]$ g++ -std=c++11 queue.cpp -o practice3
[hjkim@localhost queues]$ ./practice3 queue_input.txt c++_queue_output.txt
[hjkim@localhost queues]$ cat queue_input.txt
E 9
E 4
E 2
E 8
F
E 7
F
D
F
D
F
[hjkim@localhost queues]$ cat c++_queue_output.txt
9
9 4
9 4 2
9 4 2 8
9
9 4 2 8 7
9
4 2 8 7
4
2 8 7
2
```



```
[hjkim@localhost queues]$ python queue.py queue_input.txt python_queue_output.
txt
[hjkim@localhost queues]$ cat python_queue_output.txt
9
9 4
9 4 2
9 4 2 8
9
9 4 2 8 7
9
4 2 8 7
4
2 8 7
2
```

Submission Guideline

- Submission: **source code, your input file**
  - Where: Assignments in LMS
  - Deadline: **23:59, March. 27th (Sunday)**
- Extra points
  - **From March 28th (Monday)**
  - Share your **code, input & output** on Open Board in LMS.
  - Review classmates' code. Give questions or comments on his/her post.
  - Answer others' questions on your post.
  - Title: [Practice3]StudentID
    - e.g., [Practice3]2021000000