Data Structures

Queue Homework 1

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: Deque

- Deque is a **Double ended queue** where you can add/remove from either rear or front. It is not FIFO anymore, but provides great flexibility
- Change the **circular queue** to include
  - void enqueue_rear(int value) [same old code]
  - **void enqueue_front(int value)**
  - int dequeue_front() [same old code]
  - **int dequeue_rear()**
- Front/Rear meanings shouldn't change.
- O(1) time complexity for all methods

Problem #1: Deque

```
Deque dq(6);

dq.enqueue_front(3);
dq.enqueue_front(2);
dq.enqueue_rear(4);
dq.enqueue_front(1);
dq.display();   // 1 2 3 4
cout<<dq.dequeue_rear()<<"\n";  // 4
dq.display();   // 1 2 3
cout<<dq.dequeue_front()<<"\n"; // 1
dq.display();   // 2 3
cout<<dq.dequeue_rear()<<"\n";  // 3
cout<<dq.dequeue_front()<<"\n"; // 2
dq.enqueue_rear(7);
dq.display();   // 7
```

Problem #2: Implement a stack using a single queue

- Implement the stack functionalities but using a single queue
- Don’t implement display
  - Display is for debugging reasons
- What is the time complexity?

```
5  class Stack {
6  private:
7      Queue q;
8      int added_elements { };
```

```
Stack stk(3);
stk.push(10);
stk.push(20);
stk.push(30);

while (!stk.isEmpty()) {
    cout << stk.peek() << " ";
    stk.pop();
} // 30 20 10
```

Problem #3: Queue using 2 Stack: O(1) dequeue

- Implement Queue functionalities using 2 stack objects
- However, **dequeue**() function must remain O(1)

```
class Queue {
private:
    int size;
    int added_elements { };
    Stack s1;
    Stack s2;
```

```
Queue qu(6);

for (int i = 1; i <= 3; ++i)
    qu.enqueue(i);

cout<<qu.dequeue()<<" ";

for (int i = 4; i <= 5; ++i)
    qu.enqueue(i);

while(!qu.isEmpty())
    cout<<qu.dequeue()<<" ";
//1 2 3 4 5
```

Problem #4: Queue using 2 Stack: O(1) enqueue

- Implement Queue functionalities using 2 stack objects
- However, **enqueue**() function must remain O(1)

```
class Queue {
private:
    int size;
    int added_elements { };
    Stack s1;
    Stack s2;
```

```
Queue qu(6);

for (int i = 1; i <= 3; ++i)
    qu.enqueue(i);

cout<<qu.dequeue()<<" ";

for (int i = 4; i <= 5; ++i)
    qu.enqueue(i);

while(!qu.isEmpty())
    cout<<qu.dequeue()<<" ";
//1 2 3 4 5
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."