Weekly Homework 5

April 16, 2025

1 Stack

Given the following `Stack` definition:

```
struct NODE{
    int key;
    NODE* p_next;
};
```

```
struct Stack{
    NODE* top;
};
```

Complete the following functions to fulfill the given requirements:

1. Initialize a `Stack`:
   - `Stack* initializeStack();`
2. Push a new element onto the `Stack`:
   - `void push(Stack &s, int key);`
3. Pop the top element off the `Stack` and return its value:
   - `int pop(Stack &s);`
4. Return the number of elements in the `Stack`
   - `int size(Stack s);`
5. Return true if the `Stack` is empty
   - `bool isEmpty(Stack s);`

Output specification

Input file The input.txt file should be at the same directory level as a.exe (the executable built from your code).

```
// input.txt
init
push 1
push 2
pop
pop
pop
```

Output file The input.txt file should be at the same directory level as a.exe (the executable built from your code).

```
// output.txt
EMPTY
1
1 2
1
EMPTY
EMPTY
```

2 Queue

Given the following `Queue` definition:

```
struct NODE{
    int key;
    NODE* p_next;
};
```

```
struct Queue {
    NODE* head;
    NODE* tail;
};
```

Complete the following functions to fulfill the given requirements:

1. Initialize a `Queue`:
   - `Queue* initializeQueue();`
2. Enqueue a new element into the `Queue`:
   - `void enqueue(Queue &q, int key);`
3. Dequeue the front element from the `Queue` and return its value:
   - `int dequeue(Queue &q);`
4. Return the number of elements in the `Queue`
   - `int size(Queue q);`
5. Return true if the `Queue` is empty
   - `bool isEmpty(Queue q);`

Output specification

Input file The input.txt file should be at the same directory level as a.exe (the executable built from your code).

```
// input.tx
init
enqueue 10
enqueue 20
dequeue
dequeue
dequeue
```

Output file The output.txt file should be at the same directory level as a.exe (the executable built from your code).

```
// output.txt
EMPTY
```

```
10
10 20
20
EMPTY
EMPTY
```

3 Submission Rules

Students must adhere to the following submission guidelines:

1. Each solution must be submitted in a separate file:
   - `stack.cpp` for the stack (have main function).
   - `queue.cpp` for the queue (have main function).
2. The submission must be in a **compressed zip file** named **MSSV.zip**, containing:
   - The required C++ files. (stack.cpp, queue.cpp).
   - A `report.pdf` file describing the approach used in each solution. The image of GitHub home page to verify code is pushed on GitHub
   - Don't use $< bits/stdc++.h >$ library