Middle East Technical University | Department of Computer Engineering

CENG 301

Algorithms and Data Structures

Fall 2020-2021

Programming Assingment 2

Due date: 02.01.2021, Saturday, 23:55

1 Introduction

In this assignment, you are going to practice stack queue implementation in C++. The details of the structure are explained further in the following sections.

Keywords: *Stack, Queue, C++*

2 Problem Definition

In this assignment, the problem is to implement a bank queue management system. To solve this problem, you are required to do the followings:

1. First, you will implement the functions of the Stack class based on the linked list data structure.
2. Second, you will implement the functions of the Queue class based on the stack class that you implemented. The functions of the Queue class must be implemented using stack(s).
3. Then, you will use the Queue that you have implemented to create a bank queue management system.

Person class is provided which you must use as the elements of both stack and queue classes.

The stack & queue data structure used in this assignment are implemented as a class: Stack & Queue respectively, which contain items as the type of Person (the header is provided).

A stack class must provide the following requirements:

- Getting top person of stack (top)
- Check if the stack is empty (empty)
- Inserting (push) a person to the stack
- Deleting (pop) a person from the stack
- Display the all Person(s) in the stack (display)

A queue class must provide the following requirements:

- Adding (enqueue) a person to the queue according to the privilege level. In this function, the person must be queued according to their privilege level. The level of privilege is based on the intensity of each person's annual money exchange with the bank. The more shopping you have, the more privilege you have. 0 means not even customers, 1 means less privilege, 10 means the highest level of privilege. The privilege values range from 0 to 10.
- Deleting (dequeue) a person from the queue
- Display the all Person(s) in the queue (display)

3 Specifications

- We have provided the "person.h" header file. You are not allowed to change anything in this header.
- You will implement a linked list-based Stack class. The raw header file "stack.h" is given below. You are not allowed to change the signatures of functions.
- You will implement the Queue class using stacks (already implemented in stack.h). No other data structure is allowed. The raw header file "queue.h" is given below. You are not allowed to change signatures of functions.
- Your main duty is implementing the functions (marked by "TODO") in "stack.h" & "queue.h"
- You are given a sample main.cpp for testing purposes. Your code will not be graded with these test cases. However, test cases will be similar to the given ones.
- Do not change the signatures of the functions.
- You can write helper functions if necessary. Moreover, you can also define variables.

3.1 person.h

```
#include <string>
using namespace std;
class Person{
  private:
    string name;
    int privilege;
    Person * next;
  public:
    Person(){
      this->name = "";
      this->privilege = 0;
      this->next = NULL;
    }
    Person(string name){
      this->name = name;
      this->privilege = 0;
      this->next = NULL;
    }
    Person(string name, int privilege){
      this->name = name;
      this->privilege = privilege;
      this->next = NULL;
    }
    void setName(string name){
      this->name= name;
    }
    void setPrivilege(int privilege){
      this->privilege= privilege;
    }
    void setNextPerson(Person * next){
      this->next = next;
    }
    string getName(){
      return this->name;
    }
    int getPrivilege(){
      return this->privilege;
    }
    Person * getNextPerson(){
      return this->next;
    }


};
```

3.2 stack.h

```
#include <iostream>
#include "person.h"
using namespace std;

class Stack{

  private:
    Person * topPerson = NULL;
  public:
    //Return the top of stack
    Person * top()
    {
      //TODO
    }

    //Return whether stack is empty or not(true/false)
    bool empty()
    {
      //TODO
    }

    //Inserts the person to the stack
    void push(Person * p)
    {
      //TODO
    }

    //Extracts the topPerson and assign the next element as topPerson
    void pop()
    {
       //TODO
    }

    //Displays the person(s) in the stack
    void display()
    {
       //TODO
    }
};
```

3.3 queue.h

```
// CPP program to implement Queue using stacks (NO other data structure is allowed)
#include <string>
#include <bits/stdc++.h>
#include "stack.h"

using namespace std;

class Queue {
  private:
    Stack s1, s2;
    public:

    // Enqueue a person to the queue according to his/her privilege level
    void enQueue(Person * p)
    {
      //TODO
    }

    // Dequeue a person from the queue
    Person * deQueue()
    {
      //TODO
    }

    //Display Queue
    void display()
    {
      //TODO
    }
};
```

4 Test Case

4.1 Sample main function

```
#include "queue.h"
int main()
{
    Queue q;
    Person * p1 = new Person("ahmet");
    Person * p2 = new Person("hande", 1);
    Person * p3= new Person("gunes", 3);
    Person * p4= new Person("gurkan", 2);
    Person * p5= new Person("hakan", 5);
    q.enQueue(p1);
    q.enQueue(p2);
    q.enQueue(p3);
    q.enQueue(p4);
    q.enQueue(p5);
    cout<<"elements in the queue wrt their privileges:";
    q.display();
    cout << q.deQueue()->getName() << '\n';
    cout << q.deQueue()->getName() << '\n';
    cout << q.deQueue()->getName() << '\n';
    cout << q.deQueue()->getName() << '\n';
    cout << q.deQueue()->getName() << '\n';
    return 0;
}
```

4.2 Output

elements in the queue wrt their privileges: hakan gunes gurkan hande ahmet
hakan
gunes
gurkan
hande
ahmet

5 Regulations

1. You have to write the program in C++.
2. If your submission fails to follow the specifications or does not compile, there will be a "significant" penalty of grade reduction.
3. **Late Submission:** Late submission policy is stated in the course syllabus.
4. **Cheating:** We have zero tolerance policy for cheating. In case of cheating, all parts involved (source(s) and receiver(s)) get zero. People involved in cheating will be punished according to the university regulations.
5. **Newsgroup:** You must follow the announcements (in Odtuclass) for discussions and possible updates on a daily basis.

6 Submission

- Submission will be done via Odtuclass (odtuclass.metu.edu.tr).
- **You need to submit "e1234567.zip" where 1234567 is your student ID and it must contain the followings:**
  1. **"stack.h"**
  2. **"queue.h"**
- Do not submit a main function.
- Additional test cases may be used for evaluation.
- Only the last submission before the deadline will be graded.