# Research Hospital

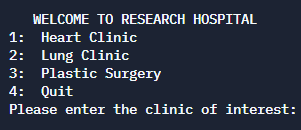
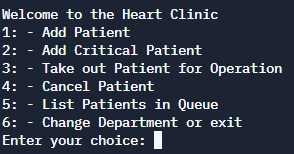
**Assignment learning objectives:**

* Group programming and task assignment
* Proper use of the list container; utilization of iterators & operations on lists
* Proper implementation of the base class
* Proper implementation of all class functions (constructors, accessors and mutators)
* Read a CSV file for basic patient information and produce errors as needed
* Implement 2 menus (1 based on clinic and 1 based on actions for the clinic)

# The assignment problem:

You have been tasked with developing a patient management system for Research Hospital. The hospital has 3 departments: Heart clinic, Lung clinic, and Plastic surgery.

This patient management system will begin the day by loading scheduled patients into the ‘queue’ as listed in the [patient.csv](https://umkc.box.com/s/2vr7yvs3897gm5fm660wgitbqtweof7j) file. Once that is complete, the application will ask the user to select the department they wish to work with (Menu 1) and provide the options for that department (menu 2)



|  |  |
| --- | --- |
| Figure 1. The application’s main screen | Figure 2. The screen of each department |

The queue for each department includes patients of the current department only. Users of your software can do the following functions within each department (see figure 2 above).

# Add new patient

The “*Add patient*” will add a new patient entry into the queue for the correct clinic. Each patient must have a first name, last name, and a social security number (cannot be empty, but can be of any length for simplicity, e.g. 123 is accepted). If the social security number is empty (return character), the operation should be canceled with cancelation message.

The maximum number of patients a department can have is 10.

Note that after adding a new patient, your application should confirm the entered data of the patient. After confirming the entered data, the user can press any key to return to the department’s menu. You do not need to ask the user if the entered data is correct or not. Just confirm it by re-printing it. Patients added should also be added to the record of patients in the transaction log (an output file that is recording all transactions by clinic – even those that error)

# 

# Add critically ill patient (emergency cases)

The “*Add critically ill patient*” will add a new patient (first name, last name, social security number) at the beginning of the current queue – passing all regular patients in the queue and after any other critically ill patients previously added. This means that patients entered the queue as critically ill will have operation before other patients, even if they entered the queue after the regular patients. Your application can add any number of patients as critically ill, but not more than 10 patients in total within each department, both regular and critically ill patients. Note that, if you have several critically ill patients then they have operations in the same order they entered the queue (First in, first out). Once all critically ill patients have been taken out to operation (removed from the queue), the application continues with the regular patients –also in a first in, first out manner. Note that after adding a new patient, your application should confirm the entered data of the patient. After confirming the entered data, the user can press any key to return to the department’s menu. You do not need to ask the user if the entered data is correct or not. Just confirm it by re-printing it. Patients added should also be added to the record of patients in the transaction log (an output file that is recording all transactions by clinic – even those that error)

# Take out patient for operation

This feature is responsible to move patients to the operation room (remove them from the queue). Of course, this should be a first-come first-served operation. Note that having critically ill patients will give them higher priority to enter the operations room (to be removed from the queue first, even if they have been added after the regular patients). After moving a patient to the operation room, your application should print the data of the moved patient (first and last name) in a new screen and on the transaction log. If no patients are left, your application should print a notification message to the screen “No more patients.” (this does not need to be recorded on the transaction log)

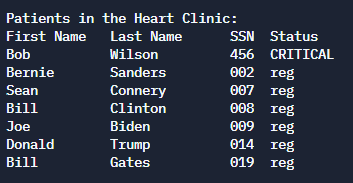


# Cancel patient

This feature will ask the user to input the social security number of the patient and then remove that patient from the queue. If such a patient does not exist, a notification message is printed to the screen “Patient Does Not Exist.” If the patient does exist, their name should be written to the screen and this information added to the transaction log.

# List all current patients

This feature lists all patients currently in the queue (first name, last name, social security number) including status (regular and critically ill patients)



# Exit the current department

This function exists the current department, clear the screen, and print the main application menu (Figure 1).

**Exit**

At the end of the day, your transaction log(s) should show:

1. The record of all patients added to each clinic at the beginning of the day
2. The record of patients added, operated on, or cancelled throughout the day along with their status (regular or critical)
3. If you end your transactions before all the queues are emptied, the queue contents should also be written to the transaction log with a note that the patient must be rescheduled.
4. Transaction logs should be grouped by clinic.

**Thoughts:**

You could use a strut called *patient* to define patients using first name, last name, Social Security number and status.

Your application must have a class called *Clinic* that includes the aforementioned functions (methods) as member functions of the class such as, AddPatient(); AddCriticalPatient(); OperatePatient(); CancelPatient(); etc. Feel free to create your own methods –these are just examples.

The use of an actual queue container may be cumbersome. You should attempt to use one of the new containers discussed if possible, along with the use of error catching as needed.

**Finally:**

As a group – you will need to submit this form: [CS201 Final Project](https://forms.gle/Bv4qvit5uyKo5jng6)

1. Source code (via repl.it) with embedded documentation including pre- and post- conditions for functions.

2. Written document explaining the role for each team member and work completed.

As an individual – you will need to submit this form: [CS201 Final Project Rubric](https://forms.gle/8zVRkB5c1c1Tfkty8)

1. Team Evaluation Rubric (this is a completion grade – your scoring will not affect your grade)