**3460:209 PROGRAMMING PROJECT 4: CS1 HOSPITAL, Bed Assignments**



**T**he CS1 hospital receives a variety of new patients each day. At various times of the day patients are held in a temporary room until there is a determination made where the patient should be placed for their visit. The final bed assignments are added at the end of the day. There also are changes and updates that have to be made. These rearrangements are reflected and placed on a grid that contains all of the current days new room assignments.

You have been hired to program the daily floor updates for the hospital using the input provided from the head nurse and a file that contains the pre-existing grid of the floors. There are 5 floors with 8 rooms on each floor. The pre-existing grid is actually a two-dimensional array that contains the following numeric codes:

1. Vacant
2. Check out
3. Occupied
4. Nurse Station
5. Transfer
6. Utility room

These indicate the actions that must take place as updates to the grid. These updates have to be made to the floors (grid) first, and then the new patients are inserted into the vacant rooms.

*Here are the basic requirements:*

* When rooms are checked out they may be filled with a transfer or with a new patient so check outs should be updated to vacancies.
* When a room is vacant it can be filled with a transfer or with a new patient.
* Transfers are to be handled first, before new patients. Transfers are to be moved to the first available room, and their existing room is eligible for new patients or for another transfer. You may have to transfer patients to a different floor.
* After all transfers are completed and check outs have been completed, new patients may begin to move in.
* Both the occupied, utility room, and the Nurse Station codes will not require any activity but they are also not available.
* When accepting the user input, you are required to validate the input and also you will need to display the total number of rooms that are currently available. You must limit the user’s input to not be greater than that value.
* The program must display the grid of rooms three times: 1) upon reading the file; 2) after the transfers and check outs but before the user input and; 3) final version of the completed grid. All codes will display as before except those that checked out (1), those will become occupied (2) or vacancies (0).

*Strategy*:

* Open the file for reading.
* Build and display the initial grid of rooms from the input file.
* Remember that room 1, floor 1, is (0,0) on the grid.
* Perform the checkout updates.
* Perform the transfer updates.
* Rebuild and redisplay the grid of rooms.
* Ask the user for their daily new patient input (display the threshold and limit the number)
* Update the new patients.
* Rebuild and redisplay the grid of rooms.

*Printing the grid of rooms*

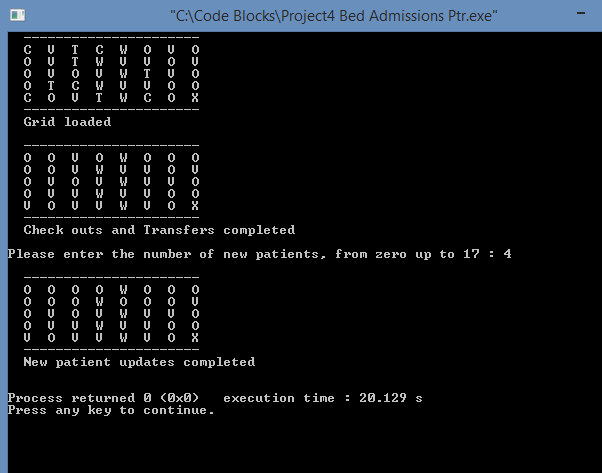
* You read in the floor grid as a 2-dimensional array of integers for the sole purpose of file reading sanity. However, this is not how it will appear when you display it on the screen.   
  + Transfers are represented by a ‘T’.
  + The Check outs are represented by a ‘C’.
  + Nurse work stations are represented by a ‘W’.
  + Occupied rooms are represented by an ‘O’.
  + A vacant room is represented by a blank ‘V’.
  + The utility rooms are represented by an ‘X’.
* There are different ways to handle this so as long as it makes sense, I won’t interfere. If you want advice, see me during office hours or ask in class.

*Code*:

* Use a two-dimensional array to represent the room grid.
* Make sure that your input from the nurse’s station is valid (it is in the range of vacancies and it must be numeric).
* At a minimum you are required to create the following functions to:
  + Read in the initial grid.
  + Display the current grid of rooms with current data on who is in what position.
  + Get the input from the nurse’s station.
  + Move the check outs.
  + Move the transfers (*important: see instructions* *below*\*).
  + Insert the new patients.
  + Calculate the number of vacancies.
* Make sure you use call by value and call by reference properly.
* Make sure you close the input file when you are finished reading it.
* Use a header comment with your name and the description of the program.
* Use pre and post condition comments for each function.
* Use appropriate code comments.
* Use appropriate variable names.
* Use constants where appropriate.
* Use good coding practices (i.e., spacing, indentation, etc.)

\*For moving transfers, your function is required to create a char array of pointers. This array will hold the addresses of the vacant positions. You will allocate an array the size of the number of vacancies and load the dynamic array with the transfers. Then move all of the transfers to available rooms (mark them as OCCUPIED) using the pointer.

Your output should look like this:



Submission:

* You will find a file called main.cpp in the Projects/Project4 folder in SVN.
* Commit your final main.cpp to SVN.
* Remember that it’s a *good* thing to commit as you develop. This is an excellent back up for you. I don’t look at the number of commits you make. I will only grade the last commit after the due date.
* **You are not permitted to work with others**. Consult with me, the graders or the tutors for help, but be prepared with actual code to work with.
* **I will not grade a commit made after 5:00 p.m. on Friday May 1st, 2015. No exceptions.**