# General Testing Guidelines

- If there is a tricky part in this program then it is likely to be the queue functions. This is particularly true when the two indices 'first' and 'last' start to wrap around the array.
- If the size of the queue array is large it will take a long time (sequence of patients' check in and check out commands) before the indices start to wrap around.
- Therefore, to test the program quickly without having to use long sequences of commands, and for the purpose of debugging only, set the size of the array to a very small value (e.g. 6). This will ensure that the indices will start to wrap around after handling a small number of patients.
- You may find it helpful for debugging to implement a command for printing the queue data structure (this is different from the 'w' command). The command displays the queue array and the position of first and last. You can call this command u (q**u**eue). Here is an example of how my queue looked like after some sequence of operations.

```
>>>u
673 (last), 777, 555 (first), 33, 783, 345,
first = 2, last = 0
```

- Once you are satisfied that the program is working correctly you can set the array size back to a reasonable value (e.g. 60) to ensure that the queue does not easily overflow during normal program operation.

# General Testing Guidelines

- You may also find it helpful for debugging to add the following command to your program (not required). You can call it **t** (status). The command displays the status of all rooms and their occupancy. Here is an example of room usage at one point during program operation.

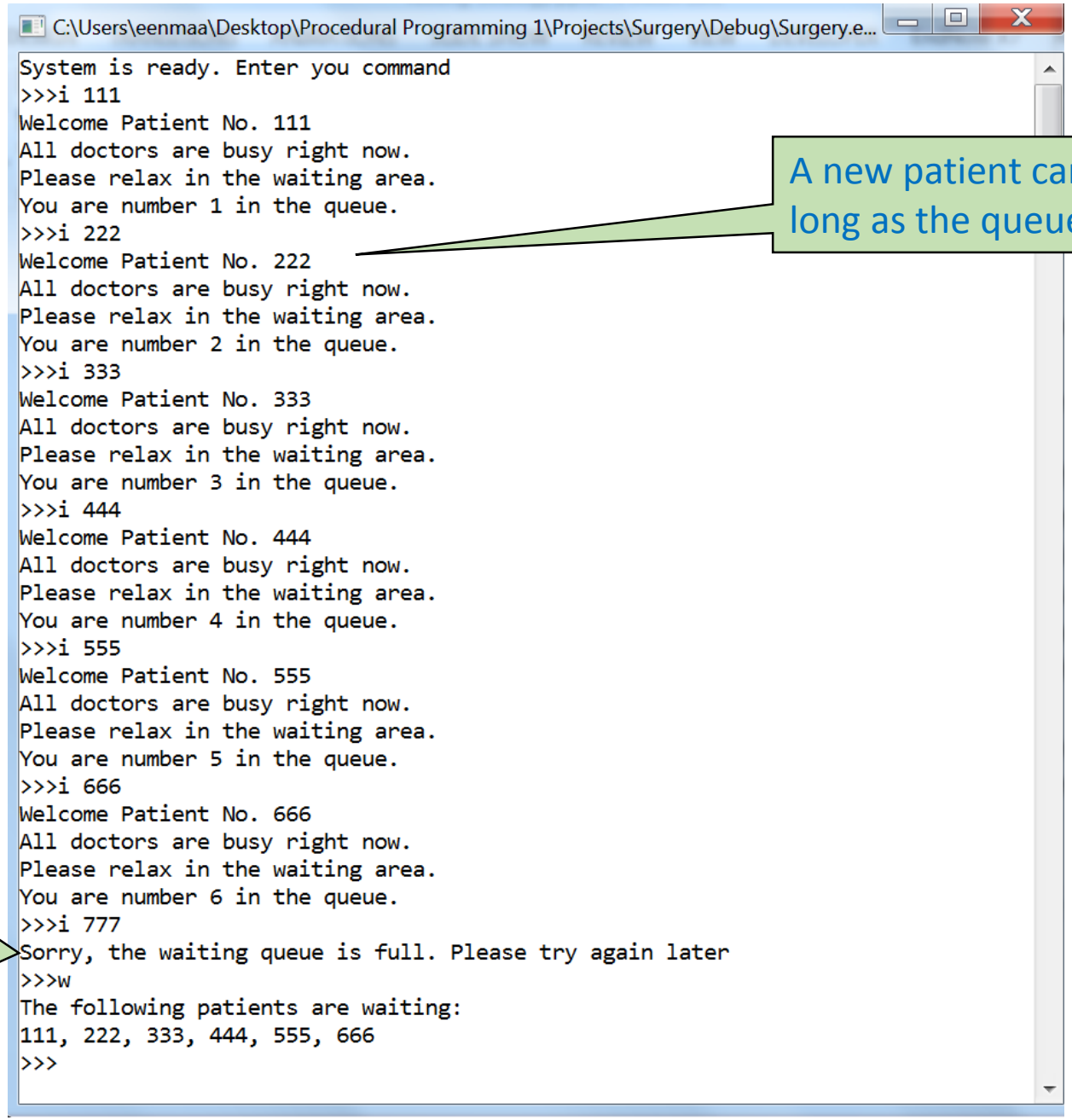
```
>>>t
Room 1: Vacant
Room 2: Vacant
Room 3: Doctor in with Patient 777
Room 4: Vacant
Room 5: Vacant
Room 6: Vacant
Room 7: Doctor in, not ready
Room 8: Vacant
Room 9: Vacant
Room 10: Vacant
>>>
```

A room can:
- be vacant, or
- has a doctor (not ready to see patients), or
- has a doctor seeing a patient.

Obviously a patient cannot be left alone in a room.

Note: to reduce clutter in the test data, the password feature for admin commands was turned off so that a user can issue any command without password verification.

# Testing the i (Patient Check in) Command




A new patient can check in as long as the queue is not full.

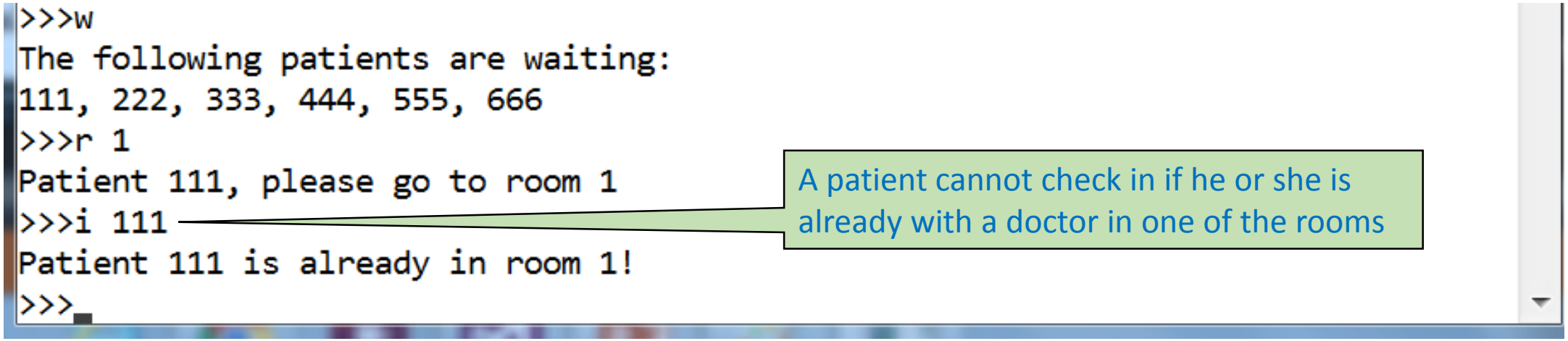
If the queue is full, the patient is asked to try again later.

# *Testing the i Command (continued)*

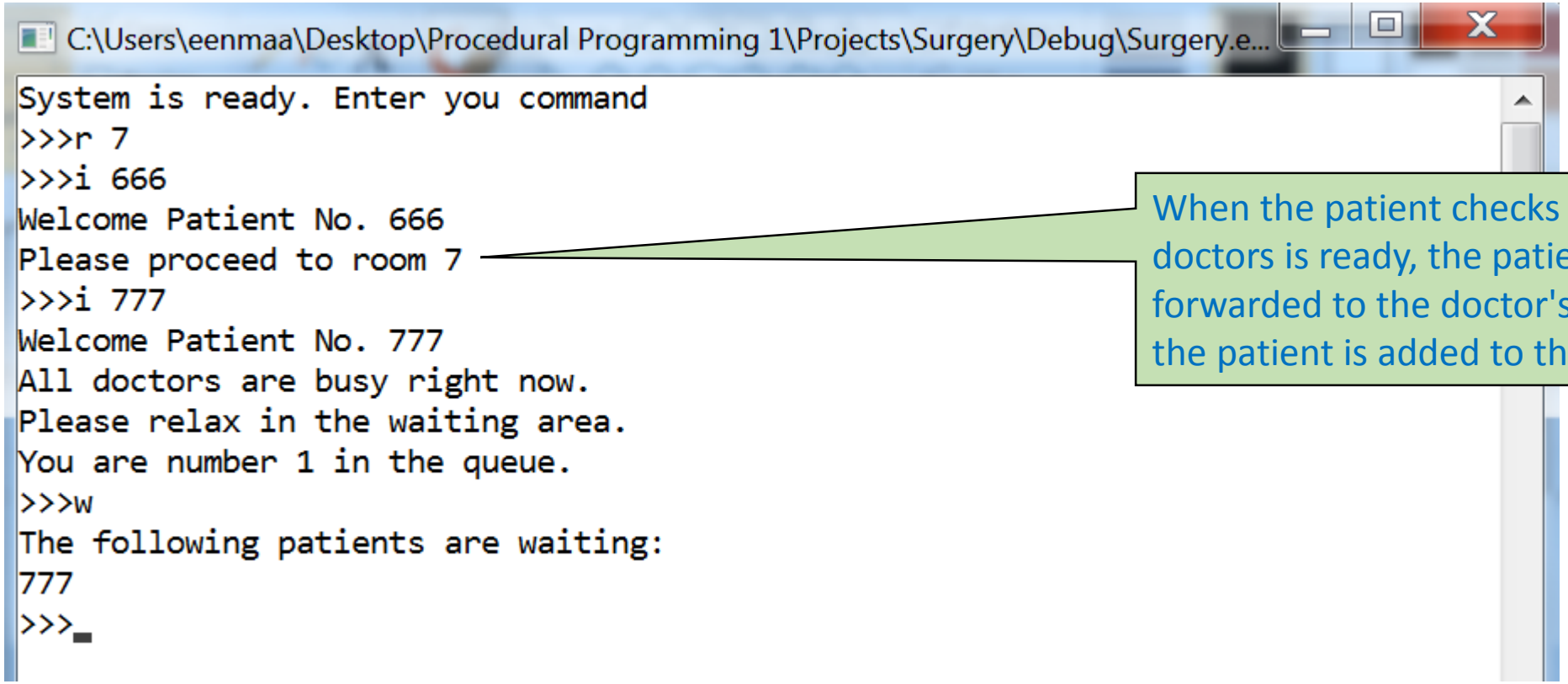
```
>>>w
The following patients are waiting:
111, 222, 333, 444, 555, 666
>>>i 555
You are already checked in
You are in position 5 in the queue
>>>i 111
You are already checked in
You are in position 1 in the queue
>>>i 666
You are already checked in
You are in position 6 in the queue
>>>
```

A patient cannot check in twice

# *Testing the i Command (continued)*



# *Testing the i Command (continued)*



# *Testing the p (Patient position) Command*

```
>>>w
The following patients are waiting:
111, 222, 333, 444, 555, 666
>>>r 1
Patient 111, please go to room 1
>>>w
The following patients are waiting:
222, 333, 444, 555, 666
>>>i 777
Welcome Patient No. 777
All doctors are busy right now.
Please relax in the waiting area.
You are number 6 in the queue.
>>>u
777 (last), 222 (first), 333, 444, 555, 666,
first = 1, last = 0
>>>p 222
You are in position 1 in the queue
>>>p 777
You are in position 6 in the queue
>>>p 999
You are not in the queue
>>>
```

If the patient is not in the queue, this should be handled as well

The position of the patient in the queue is given correctly. This position should be correct even if the queue wraps around

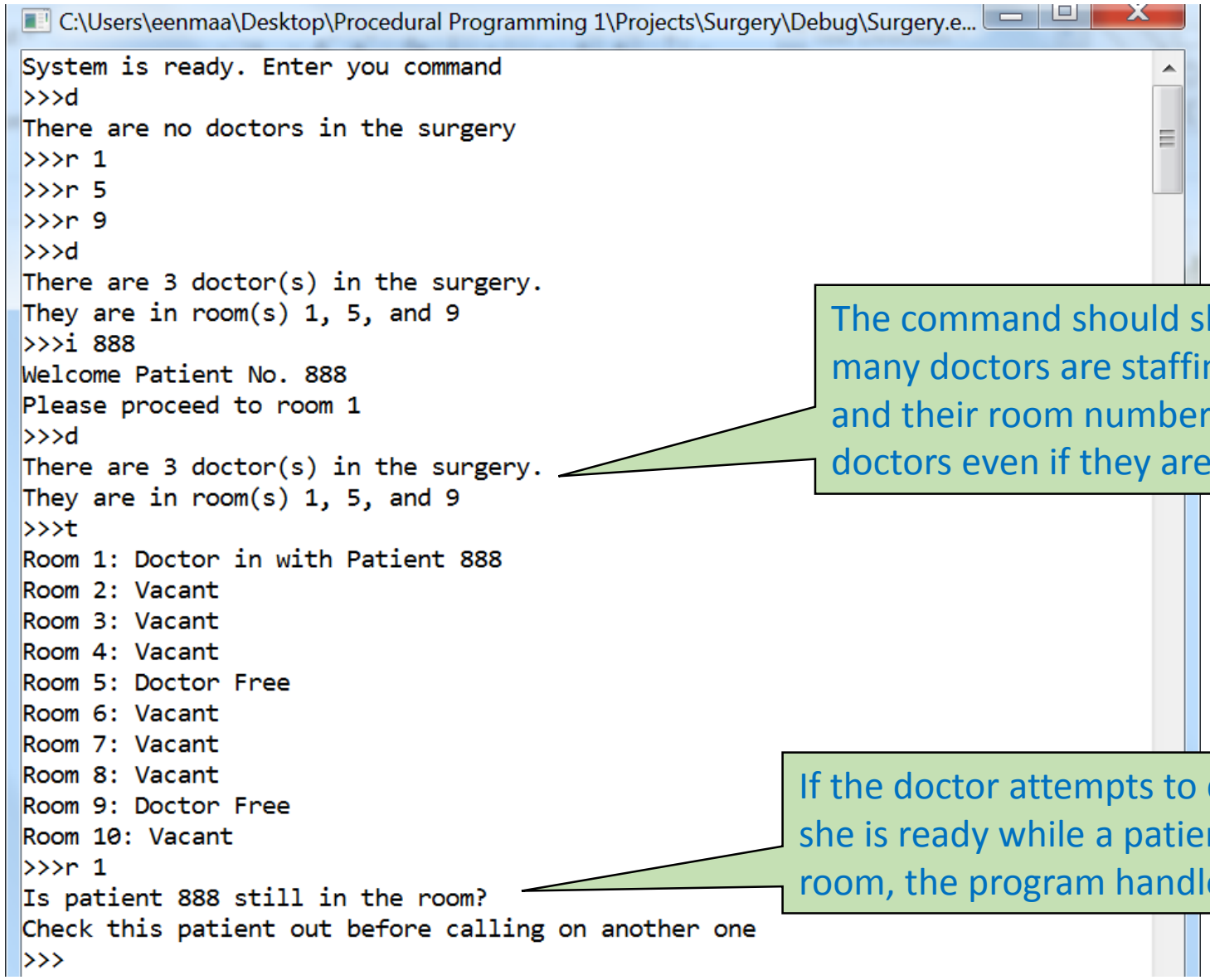
# *Testing the q (Patient quit) Command*

```
>>>u
777 (last), 222 (first), 333, 444, 555, 666,
first = 1, last = 0
>>>q 777
You have been removed from the queue!
Thank you for your visit
>>>u
777, 222 (first), 333, 444, 555, 666 (last),
first = 1, last = 5
>>>q 333
You have been removed from the queue!
Thank you for your visit
>>>u
777, 222 (first), 444, 555, 666 (last), 666,
first = 1, last = 4
>>>q 222
You have been removed from the queue!
Thank you for your visit
>>>u
777, 444 (first), 555, 666 (last), 666, 666,
first = 1, last = 3
>>>q 999
Your are not in the queue
>>>
```

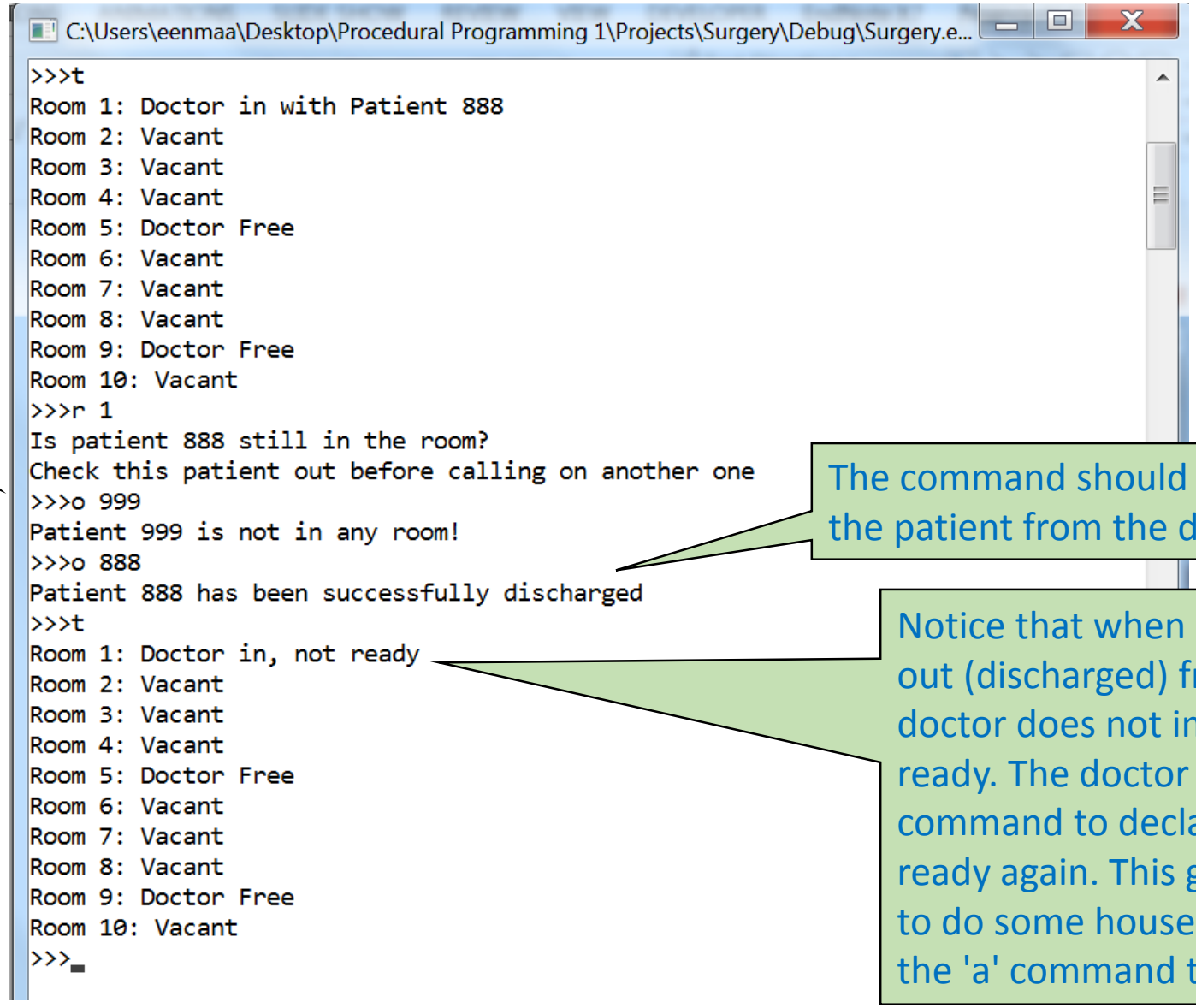
The patient should be removed from the queue correctly, whether the patient is first, last, or middle in the queue. The command should work correctly even if the queue wraps around.

Handle the case when the patient id is not in the queue

# *Testing the d (available doctors) Command*



# *Testing the o (patient check out) Command*




Handle the case when the patient id is wrong

The command should correctly discharge the patient from the doctor's room.

Notice that when a patient is checked out (discharged) from a room, the doctor does not immediately becomes ready. The doctor must use the r command to declare that he or she is ready again. This gives the doctor time to do some housekeeping tasks, or use the 'a' command to leave the room.

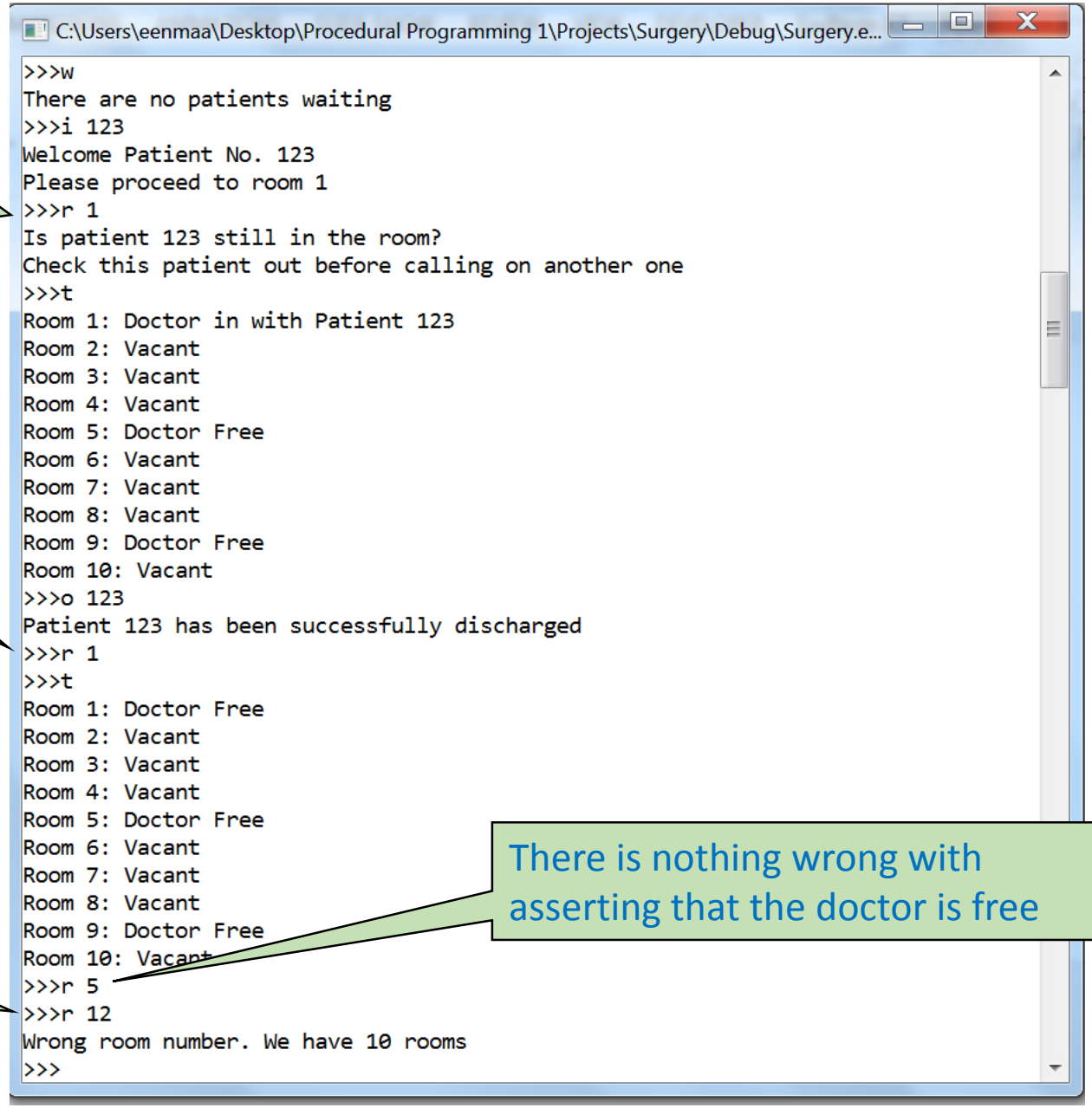
# Testing the r (doctor ready) Command

As previously noted, The doctor cannot be ready if there is a patient in the room

The doctor can use the r command now

Handle wrong room numbers.

There is nothing wrong with asserting that the doctor is free

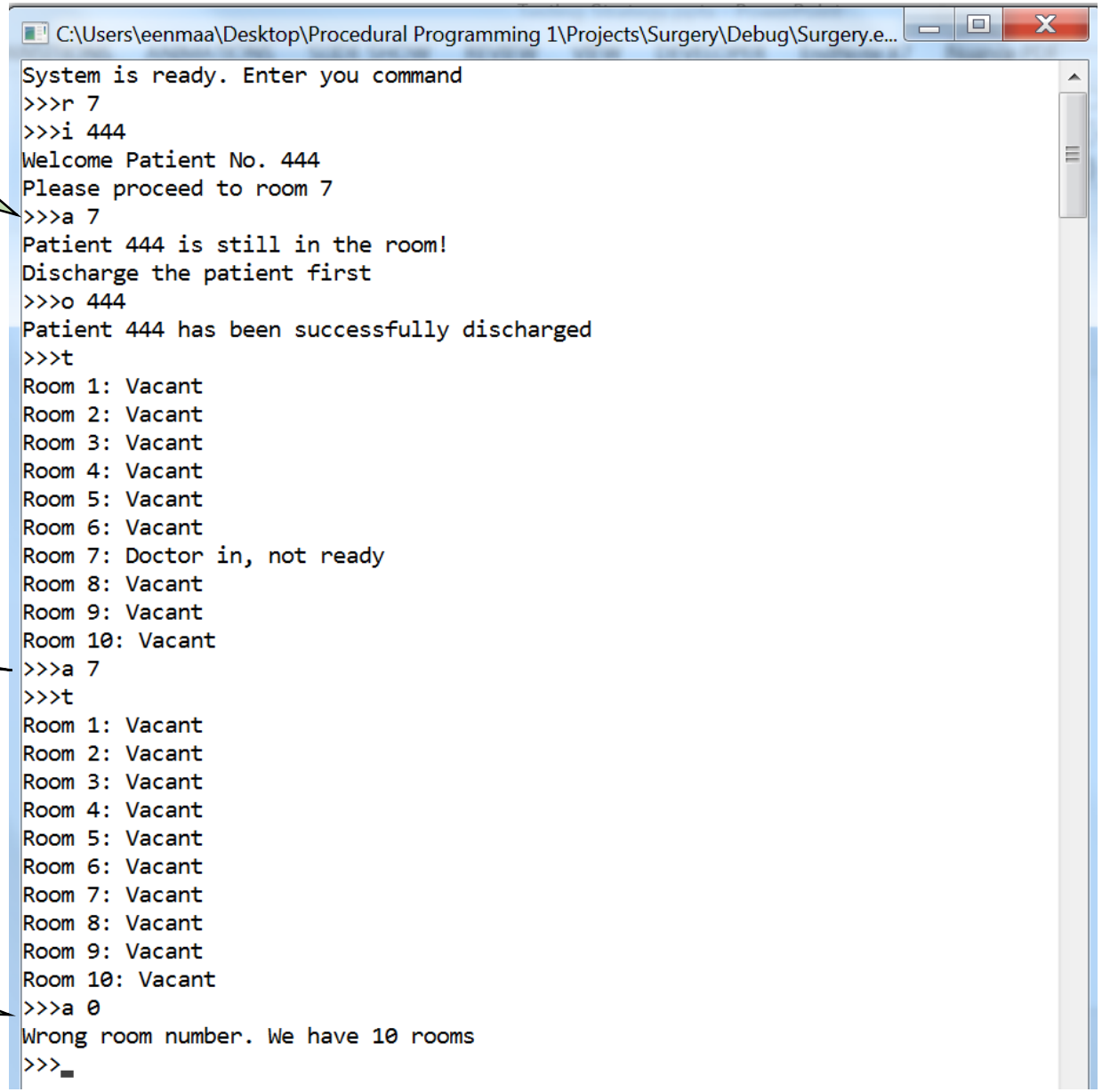


# *Testing the a (doctor away) Command*

The doctor cannot run away from a patient ☹

The doctor can run away now ☺

Handle wrong room numbers.



# *Testing the w (who is waiting) Command*

```
>>>w
The following patients are waiting:
111, 222, 333, 444, 555, 666
>>>r 5
Patient 111, please go to room 5
>>>w
The following patients are waiting:
222, 333, 444, 555, 666
>>>i 777
Welcome Patient No. 777
All doctors are busy right now.
Please relax in the waiting area.
You are number 6 in the queue.
>>>w
The following patients are waiting:
222, 333, 444, 555, 666, 777
>>>u
777 (last), 222 (first), 333, 444, 555, 666,
first = 1, last = 0
>>>
```

The waiting list (queue) should always be printed out correctly.

Notice that patient 777 is the last one in the queue even though 777 is the first element in the array.