

Due: Thursday, July 14  
Emphasis: Queues

As we mentioned in class there are many variations on the basic queue data type, including priority queues, dequeues, etc. Another example of a queue variation is the *random queue*. In a random queue, items are added to the back of the queue as normal. However, when an item is removed from the queue, instead of taking the item at the front of the queue we instead remove a random item from the queue (with equal probability of removing any item that is currently on the queue).

Modify the Queue class `remove()` method to implement a random queue. You may use either the array or linked list versions of the Queue class as your starting point. Either choice will make some things easy to do, but present some problems for other things. For example, when you randomly remove an item from the middle of an array based implementation, you need to do something about the hole left behind. For the linked list version, you need to do some work to find the item selected at random, unlike the array based method where you can directly access the item randomly chosen.

You should use a random function, like the `rand()` function in the C standard library, to choose the item to remove. The standard library function `rand()` returns a random number such that  $0 \leq \text{rand()} < 32767$ .

To convert this into a number within a specific range  $1 .. n$ , use the algorithm `rand() % n + 1`.

For instance, if we were simulating rolling dice, we would want a number in the range  $1 .. 6$ , so we could code something like: `die = rand() % 6 + 1`.

To obtain a number within the range  $0 .. n$ , use the algorithm `rand() % (n + 1)`.

The random number generator is initialized (needs to be called only once) by a call to the standard library function `srand()`. You can pass `srand` a specific value (good for testing purposes until you're sure your program is working correctly), or you can initialize it with a value from the system clock (for a random starting seed):

```
srand (237);    or    srand (unsigned (time (NULL)));
```

To use the `rand` and `srand` functions, include `stdlib.h` (or `cstdlib`), and to use the `time` function, include `time.h` (or `ctime`).

With the random queue class you just implemented, write a client that picks numbers for a lottery. Your client should put the numbers 1 through 99 on your random queue. Then it should print the results of the lottery by removing five items and displaying the winning numbers.

### Requirements

- 1) You must start with either the array or linked list versions of the Queue as presented in the book, it is your choice.
- 2) You need only modify the `dequeue()` member function to remove an item at random from the queue elements (rather than removing item from the head of the queue).

- 3) Write a small test driver, as described, that enqueues the numbers 1 to 99 then chooses 5 at random using your implementation of dequeue
- 4) Be careful that your implementation of dequeue picks a valid element, if there are only 7 items in the queue it should randomly pick one of the 7 items.
- 5) Likewise make sure that when you remove an item, you don't leave holes or accidentally lose a portion of your list of queue elements.