# Purpose

The purpose of this assignment is to obtain experience with the **Queue** and **PriorityQueue** classes, random number generation, and with a computer simulation.

# The Project

Your company has been hired by the management of a convention center in your city that is going to host a computing convention. The task your company is to address is the staffing of the registration desk. There is great anticipation for a good turnout because the convention sponsor has announced it will give out the latest smart phones and other gadgets for each person who registers on-site. Registration will be done from 8 am until 6 pm on the day before the convention begins. The convention center is planning for an expected 1,000 (random number from the **Poisson** distribution) on-site registrants who will arrive **approximately uniformly** throughout the registration period. Experience in other cities has suggests that each person takes an average of 4 minutes and 30 seconds to complete the registration process, pay the registration fees, pick up the materials, and move on. No one will be able to complete the process in less than one minute and 30 seconds. The amount of time each registrant takes should be generated from a **negative exponential probability distribution**.

Your company is to write a C# program to simulate the registration area of the convention center to determine how many registration windows must be staffed to assure that lines do not exceed 5 people in length including the person at the head of the line who is currently registering. See a [demo](http://csciwww.etsu.edu/bailes/courses/2210/Assignments/ConventionRegistration/ConventionRegistration.mp4) (an .mp4 file) or [demo2](file:///\\csciwww\bailes\Courses\2210\LectureMaterial\ConventionDemo2\conventiondemo2.mp4).

# Specifications

The program is to be flexible enough to allow for last minute changes in the number of on-site expected registrants, the expected time registration required for each registrant, the length of the time the registration process is open, and the number of registration windows that are staffed.

Create classes for **Registrants** and **Events** (**arrivals** in the registration area and the selection of the shortest line, and **departures** by people who have completed the on-site registration process). Use a **PriorityQueue** to manage the **events** in the proper order and a **List < Queue <Registrant> >** to manage the lines at the registration windows. Each registration window is modeled using a **Queue <Registrant>**. Also create a **ConventionRegistration** class that manages the simulation process for the driver.

You may assume certain things including

* The entrance doors to the convention center close at the specified time, but registrants in line at that time may complete the process.
* Incoming registrants always choose the shortest line, but once in line, they remain in the line selected.
* It is not helpful to use a clock object. All time-of-day items (starting time, closing time, etc.) should be represented by **DateTime** objects. A duration of time should be represented using a **TimeSpan** object.
* Arrival time is the time the registrant enters the registration area and chooses the shortest line. Departure time is the time at which the registrant completes the process and exits his/her line.
* A registrant may not begin registering before reaching the window at the front of his/her line.

During execution of your program, it needs to display enough graphical or pseudo-graphical information (but not just raw data) to convince a user that the simulation is legitimate and that the answers produced are realistic.

The design of the user interface and the type of C# application (**Console** or **Windows Forms**) is up to you, but it should include at least a semi-graphical display that is convincing to the staff of the convention center who are not necessarily programming literate as it runs.

# Hints/Suggestions

* The more fine-grained the timing of your simulation is (within reason), the more accurate you can make it. For example, a simulation that measures time to the nearest second is probably more accurate than one that measures time to the nearest minute which is, in turn, more accurate than one that measures time to the nearest hour or nearest day. This argument reaches a point of diminishing returns. Measuring time to the nearest femtosecond does not improve the simulation’s accuracy appreciably over measuring to the nearest second (or tenth of a second), for example.
* Each registrant is associated with two events: arrival and departure.
* Each event is associated with one registrant.
* Registrants and their arrival events may be generated using the appropriate random number generators before the simulation itself starts.
* Departure events cannot be determined easily until a registrant reaches his/her registration window during the simulation process. These events must be added to the priority queue on the fly as the program runs.
* Since you are interested in determining the number of windows that must be staffed to keep lines no longer than 5 people, you only need to consider those things that change the lengths of the lines (lines getting shorter or lines getting longer).
* In a simulation using a certain number of windows, it is worth letting the simulation run to completion even if a line exceeds 5 people.
* Because you are dealing with random numbers, a simulation with N staffed windows may fail to keep the lines sufficiently short once and succeed with the same number the next time. Once you think you know how many windows need to be staffed, make several runs with that number, with one more, and with one less to verify your conclusion.
* In writing up your conclusions, it may be useful to include some “almosts” if they are close enough. For example, if your data shows that if lines of length 6 can be tolerated for 3% of the registrants, one can staff 2 fewer windows, the convention center may value that information.
* Your display must include useful information – not just raw data. A display that just lists all events, the times they occurred, and so forth, leaving the reader to figure what the data means is not very helpful.
* The use of the **Thread.Sleep (N)** where N is the integer number of milliseconds to pause may be helpful in making your screen more readable as the application runs.

# Deliverables

In addition to your completed project, your zipped file should include a one-page or so (MS Word) report to the convention center staff describing your conclusions and recommendations. Spelling, grammar, and readability determine part of the grade on the project. You may work in groups of two on this project. Be sure that your deliverables include names of all of your team members especially in the name of the zipped file you submit.