CMPE 126 Fall 2014

Lab Exam#3: Queue Simulation

Anahit Sarao

Submitted: December 2, 2014

**Introduction**

This simulation was based on the effectiveness of slow moving trains for public transportation. The simulation has certain parameters that are fixed and other parameters that are variables and are kept track of.

Fixed:

Every five miles has a stop

People Board and Vacate Train at every stop by a number of two hundred

Distance is in Miles, Time is in Minutes

Wait time at each station is a delay of five minutes

Train travels roughly twenty-five miles per hour

Weather

Snow causes five minute delay

Rain causes three minute delay

Cloudiness causes one minute delay

Sunny has no delay

Maintenance and or Accident causes a thirty minute delay

Variable Parameters:

Stations numbers

Travel Time of Train to certain station

Number of Trains

Travel Time between stations

Number of stops so far

Using these parameters the user was asked for number of passenger’s total, distance to be traveled, randomly generated hazards and weather type. The implantation of a STL or Standard Template Library queue was initialized by the amount of trains and stations. This led to variety of results but mainly showed that the shorter the distance the more efficient the slow moving train was.

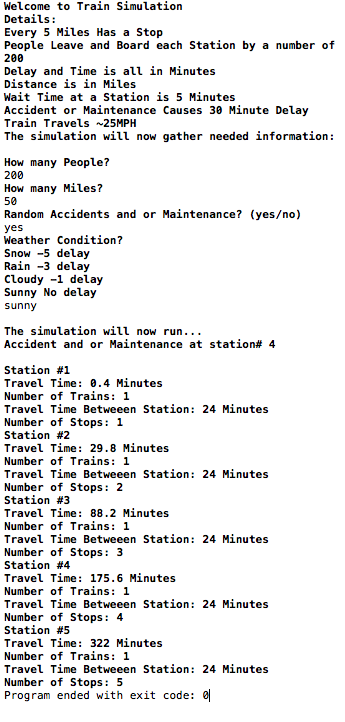
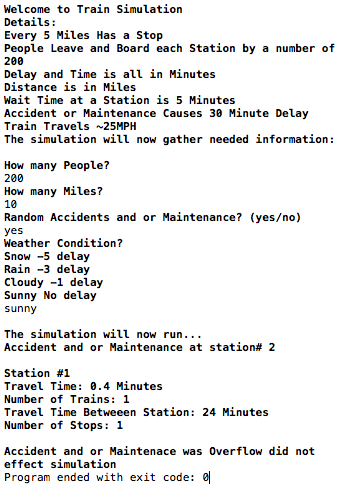
**Implementation**

Once the user had entered the basis needed information the program initialized two integer queues and one object queue. The two integer queues were needed to keep track of the number of stations and trains, while the object queue had a queue list of people waiting to be picked up at a station. User input can be seen from line 28 till line 53 while the queue initialization can be seen from line 62 till line 75 within attached file “simulation.cpp”.

Since the weather had a delay effect and there were multiple types of weather a weighting function was defined and implemented to acquire the delay amount per user choice. Function for weather weight can be seen lines 76 to line 91 within attached file “simulation.cpp”. Simulation was started as soon as all other extras had been initialized and filled with data. The simulation uses a queue of stations to keep track of how many stations or stops, the amount of delay between stations or can be the amount of time between a queues pop. Since everything is calculated before the pop it helps organize the data for each station and train, the delays for then added up within a dynamic array which had allocated memory based on the amount of stations. Once a station was popped the iterator for moved forward and new data for the station and train were accumulated. A level above the iterator are if statement checks that categorize which segment of code will run based on the parameters given by the user. This whole implementation can be thoroughly seen from line 92 till line 225 within attached file “simulation.cpp”.

**Data/Results**

Data for this simulation was randomized during the random maintenance and or accident call. This generated a number within the range of stations with a small overflow chance. Figure one shows a test run with this overflow feature. Since there was only one station to stop at and the hazard was present at the second station it did not effect this simulation. Figure two shows a result where a random hazard occurs at station four and the travel time is in increased due to the hazard delay.

****Figure 1: Output with hazard overflow Figure 2: Output with hazard no overflow

By collecting data and running the simulation countless time as shown in figure three and four a derived result was produced.

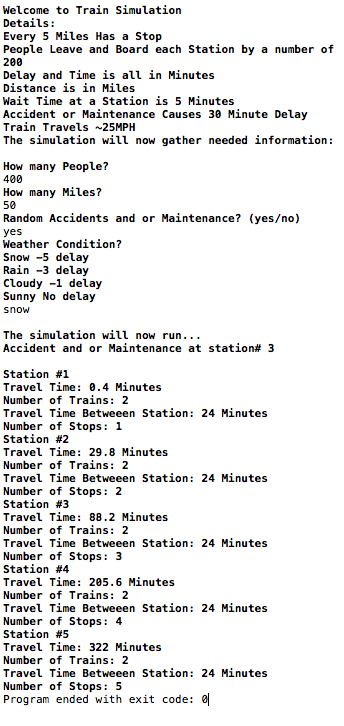
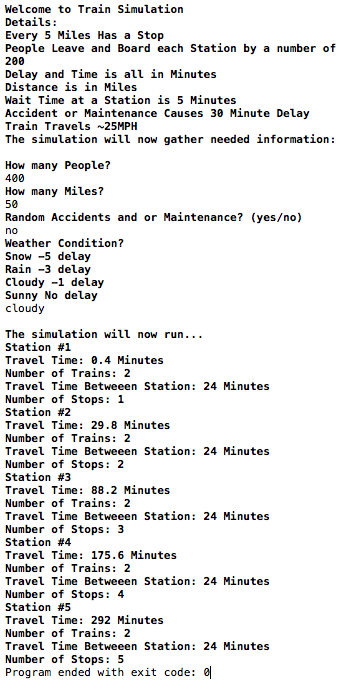
****

Figure 4: Output Result

Figure 3: Output Result

**Conclusion**

For a slow moving inner city train it is best to keep the distance even with a train that can hold two hundred passengers. This however may not be true for fast moving trains that travel from city to city, as a test was using high millage low passenger and no hazard the travel time was no efficient. For best results with a full train at 25mph a distance between 0 to 15miles is the most efficient. My code can be improved by implementing a variable speed to test different speed also to change the type of day with more passengers or less. This would improve the results and give a more real-life definitive result.