# CSCE 1101-01, 02, 05 Spring 2023 Term Project Report

# Simulation of Waiting Queues of Airplanes in an Airport

Aly Elaswad - 900225517, Ahmed Abdeen - 900225815, and Omar Mohammed - 900222116

Department of Computer Science and Engineering, AUC

*Keywords: Airplane, Waiting Queue, Queue, Linked List, Arrival, Job, Wait.*

## Abstract:

The management of airplane waiting queues is a critical task, as it requires extreme accuracy and coordination between each plane to ensure the safety of the numerous passengers daily. To simulate this process, we have developed a program that illustrates the log of events that occur during the process of landing. In addition, the program uses the qt graphical user interface to facilitate the understanding of the problem further and make the project more presentable. In this project, we will explain the findings of this simulation, define the problem, illustrate the methodology used to solve the problem, and provide our analysis of the algorithms used.

## Problem Definition

The main objective of this project is to estimate the average wait time of the airplanes remaining airborne until they land on the runway after it is free. To simulate this, we have used a double-ended queue data structure to echo the waiting queue of airplanes in real life. In this project, there is hypothetically one runway in the airport on which planes land. Each plane arrives at a random time, and it cannot land until the previous plane has finished its service time. That is why each plane has to wait before landing, and this waiting time differs from one plane to another. To help airport authorities organize flight schedules more efficiently, the average waiting time calculated from the varying waiting times of the planes is crucial. Thus, the output illustrates the realistic parameters (plane number, arrival, and service times) and finally the average waiting time which reflects the flow of airplanes through a certain airport.

## Methodologies and Algorithms

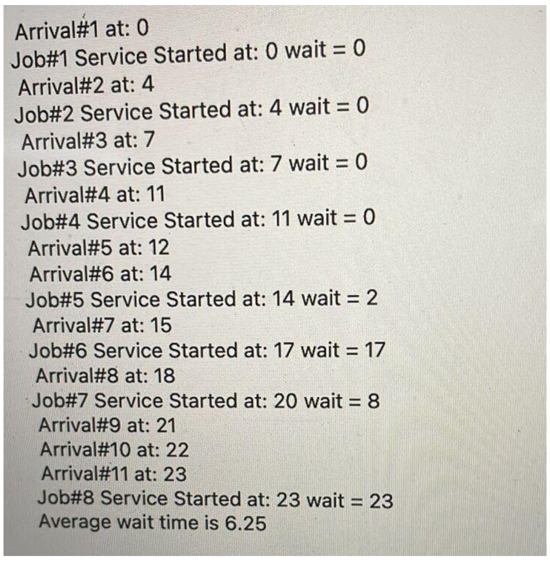
This project aimed to simulate the waiting queues of airplanes in an airport using a double-ended queue (DEQ) with linked lists. The simulation was initialized with constant variables that affected the output in wait time, number of jobs, and arrivals. The simulation did not involve any input parameters but used randomized data (Tarrival).

To complete the simulation, we followed the instructions in the presentation for Part 1 and Part 2. In Part 1, we implemented a DEQ using linked lists. We created a DEQ class, initialized the queue, and implemented functions for adding and processing nodes as required. We used a linked list to store the events in the queue and processed events in the order they were added to the queue.

In Part 2, we applied the DEQ implementation to simulate the waiting queues of airplanes in an airport. We created airplane and event structures such as RemTime, Tarrival, jobcount, wait-related variables, and ArrT<int> (vector) - these are all event-driven variables/structures that change during run time. Moreover, we initialized the simulation with constant variables and simulated the arrival of airplanes at the airport. Based on the condition of the airport (if- statement), we applied a set of guidelines to choose which aircraft could land or take off. To ensure the accuracy and validity of our simulation, we tested our implementation by comparing the output to the expected output provided in the instructions. We also ran the simulation multiple times with different constant variables to observe the effect on the output.

## Experimental Results

Throughout the process of working on this project, we have tried many code snippets on normal programs as well as graphical ones. One of the problems that we faced was that arrivals were happening, and the problem would end before it has been serviced. To solve this issue, we added a condition in the while loop to make sure that the program is not exited unless all plane arrivals have been serviced.

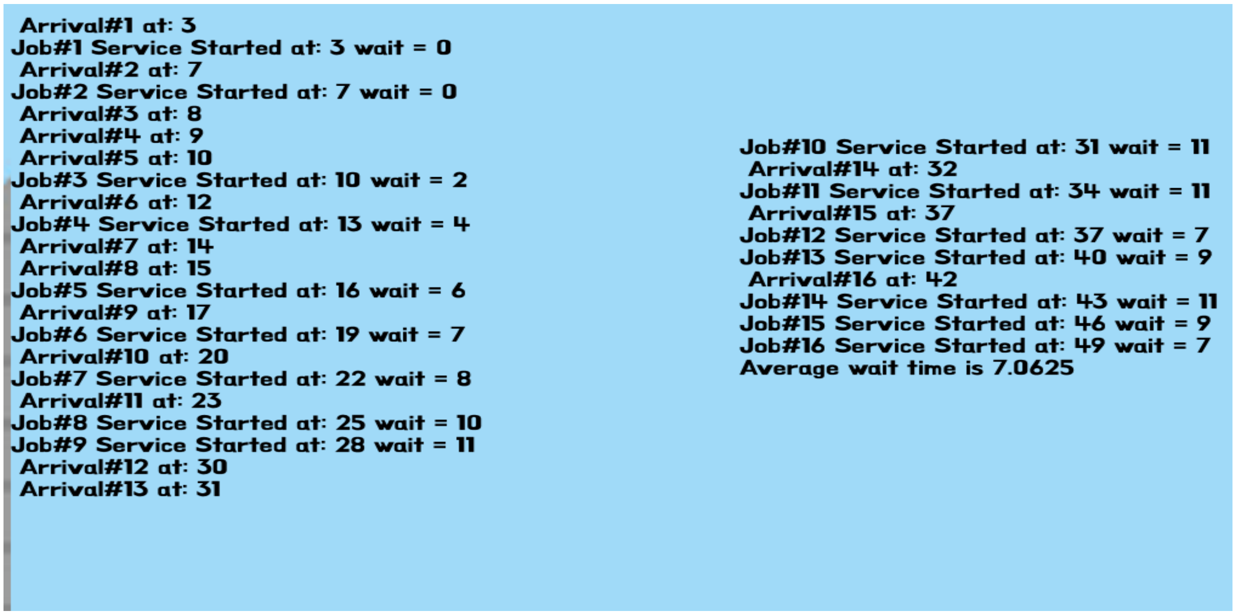


```
Arrival#1 at: 0
Job#1 Service Started at: 0 wait = 0
Arrival#2 at: 4
Job#2 Service Started at: 4 wait = 0
Arrival#3 at: 7
Job#3 Service Started at: 7 wait = 0
Arrival#4 at: 11
Job#4 Service Started at: 11 wait = 0
Arrival#5 at: 12
Arrival#6 at: 14
Job#5 Service Started at: 14 wait = 2
Arrival#7 at: 15
Job#6 Service Started at: 17 wait = 17
Arrival#8 at: 18
Job#7 Service Started at: 20 wait = 8
Arrival#9 at: 21
Arrival#10 at: 22
Arrival#11 at: 23
Job#8 Service Started at: 23 wait = 23
Average wait time is 6.25
```

*Output of the said problem*

Here, we could see that for example, Arrival #11 has no service done. This has been changed by the addition of the while condition to make it: while (t < Q.getTmax() || jobcount != i)

Another issue that could be noted from this earlier output is the inconsistency of the calculation of waitTime, we figured this could be a problem resulting from the interference of the variable value of Tarrival coming from different arrivals that may not have landed. We concluded that index parallelism is needed to obtain correct and consistent waitTime results, so we created the vector ArrT; this vector stores the Time of arrivals based on the index; this was one of the last problems solved debugging this stimulation.



```
 Arrival#1 at: 3
Job#1 Service Started at: 3 wait = 0
 Arrival#2 at: 7
Job#2 Service Started at: 7 wait = 0
 Arrival#3 at: 8
 Arrival#4 at: 9
 Arrival#5 at: 10
Job#3 Service Started at: 10 wait = 2
 Arrival#6 at: 12
Job#4 Service Started at: 13 wait = 4
 Arrival#7 at: 14
 Arrival#8 at: 15
Job#5 Service Started at: 16 wait = 6
 Arrival#9 at: 17
Job#6 Service Started at: 19 wait = 7
 Arrival#10 at: 20
Job#7 Service Started at: 22 wait = 8
 Arrival#11 at: 23
Job#8 Service Started at: 25 wait = 10
Job#9 Service Started at: 28 wait = 11
 Arrival#12 at: 30
 Arrival#13 at: 31
Job#10 Service Started at: 31 wait = 11
 Arrival#14 at: 32
Job#11 Service Started at: 34 wait = 11
 Arrival#15 at: 37
Job#12 Service Started at: 37 wait = 7
Job#13 Service Started at: 40 wait = 9
 Arrival#16 at: 42
Job#14 Service Started at: 43 wait = 11
Job#15 Service Started at: 46 wait = 9
Job#16 Service Started at: 49 wait = 7
Average wait time is 7.0625
```

In the updated (working) output, it can be seen that the wait time is actually being correctly calculated, All arrivals are also serviced.

## Analysis:

-When decreasing the Tmax value, fewer arrivals tend to happen, which justifies the logic since now the space to increment the time becomes limited.

-When decreasing the Tarrival value to 2 or 1, the average wait time seems to decrease, and rightfully so since planes have to wait less time for other planes because the earlier plane spends less time on the runway.

## Conclusions

In conclusion, the project was a successful experience considering the results and the process. It resulted in a simulation of an airplane waiting queue that resembled real life airports, as it worked with realistic variables and rules, which enhanced the project's accuracy. It can also be used as a tool for airport authorities to further optimize their flight schedules, reducing delays, postponements and other inconveniences. Reflecting on the process of finishing the project, it had mostly positive aspects. We worked as a team and divided the work accordingly, which facilitated the process. We had zoom meetings often, which increased our discussion about the project. The negative aspect was the debugging stage, as it took some time to eventually remove the bugs. However, this taught us how to work in groups effectively and provided us with some experience to use in the future.

## Acknowledgments & Reflection

We would like to acknowledge Dr. Amr Goneis's Simulation of Waiting Queue slides since it helped us quite a lot. In addition, we would like to thank Dr. Howaida Ismaeel and the TAs for providing us with enough knowledge to come up with this project, especially the hidden labs where we got hands-on experience in implementing programs like the linked list, which we used some parts of in our implementation. The youtube channel programming knowledge also helped us implement some functionalities like displaying photos on QT, which is extra information that we didn't learn in the credit lab.

We contributed evenly and accordingly to complete this project successfully. Firstly, Aly Elaswad used his coding skills and creative vision to design the visuals and buttons on the screen. He also diligently debugged the code, efficiently troubleshooting any errors that arose. Secondly, Ahmed Abdeen utilized his innovative ideas on the output and layout of the project. His extensive collaboration with Aly on QT was crucial to the project's success. Lastly, Omar Mohammed and his algorithmic expertise were critical to the success of this project. His contribution to the algorithm() function and aspects of QT was priceless. We made collective efforts to clean up the program and ensure its modularity.

## References

**https://www.youtube.com/results?search_query=photos+on+qt**
**-Provided files on BB.**