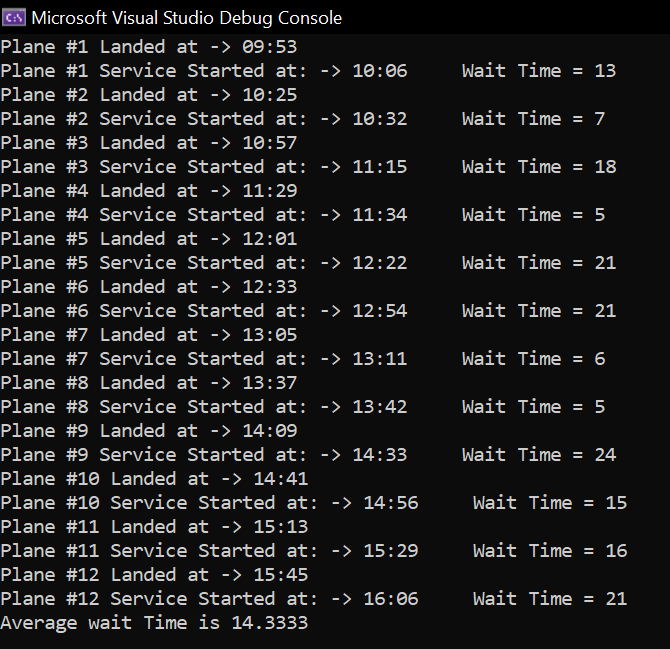
Group Members:

Malek Mahmoud 900222057

Marwan Abudaif 900221725

Mazin Bersy 900221972

Below are three randomized outputs in a row:



A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

The Projects contains 6 files:

1. Queue. H : A template queue class responsible to enqueue and dequeue airplanes in a first in first out manner

2. Plane.h: contains the declaration of variables and list of functions in the plane class

3. Plane.cpp: Contains definitions of the functions in the .h file

4. Airport.h: contains the declaration of variables and list of functions in the airport class

5. Airport.cpp: Holds the definitions of the functions in the .h file for airport

6.Airport system.cpp: This file serves as our main file and # includes all the past ones

**Queue.h:**

It uses a template double linked list, which means it has pointers pointing backward and forward for all nodes.

isEmpty(): checks if the array is empty

insertRear(T n): inserts a node at the end

getcounter(): returns how many nodes exist in the array

insertFront(T n): inserts node at the front

insertAfter(T v, Node<T>\* curr): inserts node after a specified node given in the parameter

removeRear(): removes node at the end of the array

removeFront(): removes node at the begginign of the array

viewFront(): returns value at the first node

viewRear(): returns value at the end of the node

length(): returns current size of queue

print(): prints arrival time for the plans in the queue

isFull(int t): checks if the queue is full

sort(): calls quicksort to sort the array by comparing arrival times

dealWithClashes(): deals with any clashes between arrival times within 10 mins

**Plane.h:**

static int created;

static int baseTime;

static because we need the value to be carried on to the latest made objects, rather than always starting from zero every time the constructor is called

bool operator>(const Plane& other) const {

return this->arrivalTime > other.arrivalTime;

}

Overloading operator to be used to compare arrival times in planes

**Plane.cpp:**

Plane(): If it is the first plane, it sets its arrival time at any random time and saves it as basetime

If not then it sets an arrival time within a specified range not too far away from the basetime and adds that number to the basetime

PrintArrivalTime(): Prints arrival time for the plane

getWaitTime(): randomizes a wait time from 5 to 30 minutes

**Airport.cpp:**

service(): is used in run sim in order to print time landed for each plane, its wait time, and what time service starts at

averageWait(): total weight time / by number of planes

PopulateAirQueue(): makes a randomized sized dynamic array of planes and enters them in a queue

Then it sorts it

Deals with clashes

And sets time and basetime equal to the first arrival time

runsim(): calls the service function then displays average wait time