

# CMPN102-Project-S2022.docx

Data Structures and Algorithms

Project Requirements

Cairo University		
Faculty of Engineering		Spring 2022
Computer Engineering Department		
CMP N102		

*Data Structures and Algorithms*



***Shipping Company***

*Project Requirements*

**Objectives**

By the end of this project, the student should be able to:

Understand unstructured, natural language problem description and derive an appropriate design.

Intuitively modularize a design into independent components and divide these components among team members.

Build and use data structures to implement the proposed design.

Write a **complete object-oriented C++ program** that performs a non-trivial task.

## Introduction

A shipping company needs to handle cargo delivery the most efficient and profitable way. The company needs to automate the cargo-truck assignment process to achieve good and fair use of its trucks. Fortunately, the company manager discovers your programming skills and your deep knowledge of different data structures and decides to hire you to develop a program that **simulates** the **operation of the** cargo delivery process and **calculates** some related **statistics** in order to help improve the overall process.

## Project Phases

<i>Project Phase</i>	<i>%</i>	<i>Deadline</i>
Phase 1	30%	Week 9
Phase 2	70%	Week 14

**Late Submission is not allowed**

***NOTE: Number of students per team = 4 students.***

**The project code must be totally yours. The penalty of cheating any part of the project from any other source is not ONLY taking ZERO in the project grade but also taking MINUS FIVE (-5) from**

**other class work grades. So it is better to deliver an incomplete project rather than cheating it. It is totally your responsibility to keep your code away from others.**

## **Cargos and Trucks**

### **Cargos:**

The following pieces of information are available for each cargo:

**Preparation (Ready) Time:** the Time (day:hour) when the cargo is ready to be assigned to a truck.

**Load/Unload Time:** Time (in hours) to load/unload the cargo to/from the truck.

**Cargo Type:** There are 3 types of cargos: VIP, Special and Normal cargos.

**VIP cargos** must be assigned first before other cargos.

**Special cargos** are cargos to need special trucks like frozen cargos or chemical cargos,... etc

**Normal cargos:** all other cargos

**Delivery Distance:** The distance (in kilometers) from the company to the delivery location of the cargo.

**Cost:** The cost of delivering the cargo.

### **Trucks:**

At startup, the system loads (from a file) information about the available **trucks**. For each truck, the system will load the following information:

**Truck Type:** There are 3 types of trucks: VIP trucks, Normal trucks, and Special trucks.

**VIP trucks (VT)** are trucks that are used basically for VIP cargos.

**Special trucks (ST)** are trucks that are equipped to carry special cargos.

### **Normal trucks (NT)**

**Truck Capacity (TC):** A truck can deliver many cargos in the same journey. TC is the number of cargos a truck can carry to be fully loaded. (TC is read from input file).

**Maintenance (Checkup) time:** The duration (in hours) of checkups/maintenance that a truck needs to perform after finishing **J** delivery journeys. (J is read from input file)

**Speed:** in kilometers/hour.

**Delivery interval (DI):** The time a truck takes to deliver all its cargos. and come back to the company. It can be calculated as:

*DI = (Delivery Distance of furthest cargo)/truck speed + Sum of unload times of all its cargos + time to come back*

### **Cargo Assignment Criteria**

To determine the next cargo to assign (if a truck is available), the following **assignment criteria** should be applied for all the ready un-assigned cargos **on each hour**:

#### **Important – Basic Assignment Rules**

The Company working hours are from 5:00 to 23:00. Otherwise, it is **off-hours**.

New Assignment can be done **ONLY** during working hours.

The allowed activities during **off-hours** are: truck maintenance and trucks moving for delivery or returning back to the company.

All trucks of the same type have the same capacity.

**Loading Rule:** If some cargos are ready and waiting to be assigned to a truck, **don't** start loading cargos as long as the number of ready cargos is less than capacity of the first available truck (TC).

For example: if you have 3 ready normal cargos and you have an available normal truck but with  $TC=4$ , don't load those cargos until a 4th cargo is ready. Only then start loading the cargos. This rule may be ignored in favor of Maximum Wait Rule (See next point).

**Maximum Wait Rule:** If there is an available truck that is suitable for a cargo that has been waiting for **MaxW** hours (or more), such cargo should be immediately loaded and moved to its destination. MaxW is loaded from input file. Ignore "Loading Rule" in this case.

A truck **can't** be loaded with more than one cargo type in the same journey.

### **Assignment Order: (if TC cargos of certain type are ready)**

1. First, assign **VIP cargos** to ANY available truck of any type. However, there is a priority based on the truck type: first choose from VIP trucks THEN normal trucks THEN special trucks. This means that we do not use normal trucks unless all VIP trucks are out, and we do not use special trucks unless trucks of all other types are out.
2. Second, assign **special cargos** using the available special trucks **ONLY**. If no special truck is available, wait until one comes back.
3. Third, assign **normal cargos** using any type of trucks *EXCEPT special trucks*. First use the available normal trucks THEN VIP trucks (if all normal trucks are out).
4. If a cargo cannot be assigned on the current hour, it should wait for the next hour. On the next hour, it should be checked whether the cargo can be assigned now or not. If not, it should

wait again and so on.

**NOTES:** If cargos of a specific type cannot be assigned on the current hour, try to assign the other types (e.g. if special cargos cannot be assigned on the current hour, this does NOT mean not to assign the normal cargos).

This is how we prioritize the assignment of cargos of different types, but how will we prioritize the assignment of cargos of **the same type**?

**For special and normal cargos**, assign them based on a first-come first-serve basis. Cargos that are ready first are assigned first.

**For VIP cargos**, you should design a priority equation for deciding which of the available VIP cargos should be assigned first. VIP cargos with a higher priority are the ones to be assigned first.

☞ You should develop a reasonable **weighted** priority equation depending on at least the following factors: *the cargo preparation time, delivery distance, the cargo's cost.*

There are some additional services that the company has to accommodate:

**For normal cargos ONLY**, a request can be issued to **promote** the cargo to become a VIP one. A request of cargo **cancellation** could also be issued.

A Cargo that has been already loaded to a truck cannot be canceled or promoted.

**For normal cargos ONLY**, if a cargo waits more than **AutoP** days from its preparation time to be assigned to a truck, it should be **automatically promoted** to be an VIP cargo. (**AutoP** is read from

the input file).

## **Simulation Approach & Assumptions**

Simulation time is always expressed in **Day:Hour**. Simulate the changes in the system every **hour**. Every 24 hours, increment the days.

### **Some Definitions**

**Preparation Time (PT):** The time (day:hour) at which the cargo is ready to be assigned to a truck.

**Move Time (MT):** The time at which the truck carrying the cargo starts to move to deliver the cargo.

**Waiting Cargo:** The cargo that is ready (i.e. cargo's PT < current time) but the cargo is not loaded to a truck yet or loaded but the truck doesn't move yet. At each hour, you should choose the cargo(s) to assign from the unloaded cargos.

**Waiting Time (WT):** The time from the preparation of a cargo until its truck starts to move to deliver it.

$$WT = MT - PT$$

**moving Cargo:** The cargo that is on its way but is not delivered yet.

**Delivered Cargo:** The cargo that has been delivered.

### **Cargo Delivery Time (CDT):**

Once a cargo reaches its destination, it should be unloaded from the truck. The truck then moves on to deliver the remaining cargos or returns back to the company if this is the last cargo it has.

$$CDT = MT + \text{Cargo Distance/Truck Speed} + \text{Cargo Unload Time}$$

**Truck Active Time:** Time a truck is loading or in delivering cargos. This doesn't include time for a truck to return after delivery.

**Truck utilization:** (percentage)

$$= \text{tDC}/(\text{TC} * \text{N}) * (\text{tAT}/\text{TSim}) , N \neq 0 \text{ (if } N=0, \text{ utilization} = 0\%)$$

tDC: total cargos delivered by this truck, TC: truck capacity,

N:total delivery journeys of this truck

tAT: total truck active time

TSim: total Simulation time

## **Assumptions**

If the truck is available on time T, it can be loaded with cargo starting from that time according to cargo assignment criteria explained above.

More than one cargo can be ready at the same time. Also, more than one cargo can be assigned to different trucks at the same time as long as there are available trucks.

A truck cannot be assigned a cargo during its checkup time

Checkup duration and truck speed are the same for all trucks of the same type.

## **Input/Output File Formats**

Your program should receive all information to be simulated from an input file and produce an output file that contains some information and statistics. This section describes the format of both input and output files and gives a sample for each.



## The Input File

First line contains three integers. Each integer represents the total number of trucks of each type.

**N:** for normal trucks

**S:** for special trucks

**V:** for VIP trucks

The 2nd line contains three integers for trucks speed:

**NS:** is the speed of all normal trucks (kilometers/hour)

**SS:** is the speed of all special trucks (kilometers/hour)

**VS:** is the speed of all VIP trucks (kilometers/hour)

The 3rd line contains three integers for trucks capacities:

**NTC:** Normal trucks capacity (no. of cargos)

**STC:** Special trucks capacity (no. of cargos)

**VTC:** VIP trucks capacity (no. of cargos)

The 4th line contains four integers:

**J:** is the number of journeys the truck completes before performing a checkup

**CN:** is the checkup duration in hours for normal trucks

**CS:** is the checkup duration in hours for special trucks

**CV:** is the checkup duration in hours for VIP trucks

Then a line with two integers **AutoP** and **MaxW**

**AutoP** represents the number of hours after which a normal cargo is automatically promoted to a VIP cargo.

**MaxW** : see its definition in Cargo Assignment Criteria section above

The next line contains a number **E** which represents the number of **events** following this line.

Then the input file contains **E** lines (one line for **each event**). An event can be:

Ready event of a new cargo. Denoted by letter **R**, or

Cancellation of an existing cargo. Denoted by letter **X**, or

Promotion of a cargo to be an VIP cargo. Denoted by letter **P**.

**NOTE:** The input lines of all events are **sorted by the event time in ascending** order.

## Events

**Ready event line** has the following information:

**R** (letter R at the beginning of the line) means a cargo ready event.

**TYP** is the cargo type (*N: normal, S: special, V: VIP*).

**ET** is the event time.

**ID** is a unique sequence number that identifies each cargo.

**DIST** is the cargo distance (in kilometers)

**LT** Time (in hours) to load or unload the cargo.

**Cost** is the cargo cost.

**Cancellation event line** has the following information:

**X** (Letter X) means a cargo cancellation event.

**ED** is the event time.

**ID** is the ID of the cargo to be canceled. This ID must be of a normal cargo.

**Promotion event line** has the following information:

**P** (Letter P) means a cargo promotion event.

**ED** is the event time.

**ID** is the ID of the cargo to be promoted to VIP. This ID must be of a normal cargo.

**ExtraMoney:** Extra money for promotion

### Sample Input File

```
3 3 2          ≡ no. of trucks of each type (N, S, V)
80 70 140     ≡ truck speeds of each type (km/h)
6 4 2         ≡ Capacity of each truck type (N, S, V)
12 9 8 7 ≡ no. of journeys before checkup and the
checkup durations
10 20        ≡ auto promotion limit (days), MaxW (hours)
8            ≡ no. of events in this file
R N 1:3 1 200 2 500 ≡ Ready event example
R N 1:10 2 25 1 413
R S 1:10 3 50 2 356
R V 2:22 4 90 3 400
X 4:12 1 ≡ cancellation event example
R N 5:7 5 56 2 187
P 9:3 2 200 ≡ promotion event example
R V 11:6 6 19 3 1006
```

### The Output File

The output file you are required to produce should contain **M** output lines (a line for each delivered cargo) of the following format:

**CDT CID PT WT TID**

which means that the cargo identified by id=**CID** has been ready at

time **PT**. It then waited for a period **WT** to move and finally delivered at time **CDT** by truck of id=**TID**.

*(Read the "Definitions Section" mentioned above)*

The output lines **must be sorted** by **CDT** in ascending order. If more than one cargo is delivered at the same time, **store them in any order**.

Then the following statistics should be shown at the end of the file:

1. Total number of cargos and number of cargos of each type
2. Average waiting time for all cargos
3. Percentage of automatically-promoted cargos (relative to the total number of normal cargos)
4. Total number of trucks and number of trucks of each type
5. Average active time for all trucks
6. Average utilization for all trucks

### Sample Output File

The following numbers are just for clarification and are not produced by actual calculations.

<b>CDT</b>	<b>ID</b>	<b>PT</b>	<b>WT</b>	<b>TID</b>
<b>18:3</b>	<b>1</b>	<b>7:4</b>	<b>0:9</b>	<b>3</b>
<b>20:5</b>	<b>10</b>	<b>2:10</b>	<b>12:5</b>	<b>5</b>
<b>20:5</b>	<b>4</b>	<b>15:1</b>	<b>3:6</b>	<b>15</b>
.....				
.....				
Cargos: 124		[N: 100, S: 15, V: 9]		
Cargo Avg Wait = 3:10				

Auto-promoted Caros: 4%

Trucks: 20 [N: 13, S: 5, V: 2]

Avg Active time = 91%

Avg utilization = 87%

## Program Interface

The program can run in one of three modes: **interactive**, **step-by-step**, or **silent mode**. When the program runs, it should ask the user to select the program mode.

**1. Interactive Mode:** Allows user to monitor the cargos and trucks. The program should print an output like that shown below. In this mode, the program prints the current time then pauses for an input from the user ("Enter" key for example) to display the output of the next time.

**Current Time (Day:Hour):70:9**

**7 Waiting Cargos: [110,113] (116,118) {119,112,114}**

-----

**4 Loading Trucks: 5[3,4,12,15] 1(6) 7{9,11} 2[5,7]**

-----

**4 Empty Trucks: [4], (10), [6], {8}**

-----

**5 Moving Cargos: 7[45, 52] 12{34, 77, 80}**

-----

**2 In-Checkup Trucks: [2] {3}**

-----

3 Delivered Cargos: {5} [4] (1)

## Output Screen Explanation

The numbers shown are the IDs of cargos and trucks printed according to their types. We use [ ] with **Normal cargos and trucks**. We use ( ) with **special** ones and { } with **VIP**.

In line 3 for example, a notation like **7{9,11} 2[5,7]** means truck 7 is being loaded by VIP cargos 9,11 and truck 2 is being loaded by normal cargos 5,7.

The above screen is just for explanation and is not generated by actual simulation.

**2. Step-By-Step Mode** is identical to the interactive mode except that after each hour, the program waits for one second (not for user input) then resumes automatically.

**3. Silent Mode**, the program produces only an output file (See the "File Formats" section). It does not print any simulation steps on the console. It just prints the following screen

**Silent Mode**

**Simulation Starts...**

**Simulation ends, Output file created**

**NOTE:** No matter what mode of operation your program is running in, **the output file** should be produced.

## Project Phases

You are required to write **object-oriented** code and use classes to implement different data structures in your system.

**Before explaining the requirement of each phase, all the**

**following are NOT allowed to be used in your project:**

You are not allowed to use **C++ STL** or any external resource that implements the data structures you use. ***This is a data structures course where you should build data structures yourself from scratch.***

You need to get instructor's approval before making any **custom (new)** data structure.

**NOTE:** No approval is needed to use the known data structures.

***Do NOT allocate the same Cargo more than once.*** Allocate it once and make whatever data structures you chose point to it (use pointers). Then, when another list needs an access to the same cargo, DON'T create a new copy of the same cargo; just **share** it by making the new list point to it or **move** it from current list to the new one.

**SHARE, MOVE, DON'T COPY...**

You are not allowed to use **global variables** in your code.

You need to get instructor approval before using **friendships**.

## **Phase 1**

In this phase you should decide the data structures that you will use in your project.

Selecting the appropriate DS for each list is the core target of phase 1 and the project as a whole.

***Phase1 Report*** You should deliver a report in the following format:

--	--	--

Cairo University, Faculty of Engineering

Computer Engineering Department

Data Structures and Algorithms

Spring 2022

## Data Structures and Algorithms

### Project Phase1 Report

Team Name:

Number of members:

Team Email:

#### Members' Info:

Member Name	ID	Email

#### Project Data Structures

List Name	Chosen DS	Justification
e.g. Events List	Write the DS you have chosen  e.g.	Write here the <b>operations</b> you need to perform on this list and the <b>complexity</b> of each operation  e.g.
or	Queue/Stack/	Insert a cargo = $O(1)$
Waiting Cargos	Pri-Q/ Tree ..  etc	Remove cargo = $O(\log n)$  ....etc

**☰ Repeat the above for each list in the project**

You should cover all lists in your project

See "**Data Structures Selection Guidelines**" before writing this report

#### Data Structures Selection Guidelines:

1. Do you need a separate list for each cargo type? Cargo assignment criteria described above should affect your



decision.

2. Do you need a separate list for each truck type?
3. Do you need a separate list for each cargo status. i.e. a list for waiting and another one for moving and a third one for delivered or just one list for all?
4. Do you need a separate list for each truck status. For example, would you make a list for available trucks and another one for those under checkup,...etc. or just one list for all.
5. Do you need to store **delivered** cargos? When should you get rid of them to save memory?
6. **Which list type** is much suitable to represent the lists taking into account the **complexity of the operations** needed for each list (e.g. insert, delete, retrieve, shift, sort, etc.). You may need to make a survey about the complexity of the operations. Then, decide what are the most frequent operations needed according to the project description. Then, for this list, choose the DS with best complexity regarding those frequent operations.
7. Keep in mind that you are **NOT** selecting the DS that would **work in phase1. You should choose the DS that would work efficiently for both phases.**
8. **Important:** if you find out that a list can be implemented using two types, you should choose the more restricted type. For example if a list can be implemented using queue and a general list, you should use queue.

**Note:** You need to read "*File Format*" section and to see how the input data and output data are sorted in each file because this will affect your DS selection.

### ***Phase1 Code***

***You should to finalize the following in phase1:***

1. **Full data members** of classes (See Appendix A below)
2. **Full implementation** for all data structures that you will use to represent the lists of cargos and trucks.
3. File loading function.
4. ***Simple Simulator function for Phase 1.*** The main purpose of this function is to test different lists and make sure all operations on them are working properly. This function should:

1. Perform any needed initializations
2. Call file loading function
3. At each timestep do the following:
4. Get the events that should be executed at current timestep
  1. For Preparation event, generate a cargo and add it to the appropriate waiting cargos list.
  2. For cancellation event, delete the corresponding normal cargo (if found)
  3. For promotion event, move cargo (if found) from normal to VIP.
5. Pick one cargo from each cargo type and move it to moving cargo list(s).

**Note 1:** the cargo you choose to delete from each type must be the first cargo that should be assigned to an available truck in phase 2.

**Note 2:** NO actual cargo-truck assignment is required in Phase 1.

6. Each 5 timesteps, move a cargo of each type from moving-cargo list(s) to delivered list(s)
7. Print all info to the interface as described in "Program Interface" section without truck info.

***The simulation function stops when there are no more events and all cargos are in delivered list(s).***

## **Phase1 Video**

*You should record a short video (about 3 minutes) showing a demo for phase1*

### **Phase 1 Deliverables:**

Each team is required to submit:

A text file named **ID.txt** containing team members' names, IDs, and emails.

#### **Phase1 Report**

**Phase1 code** [Do not include executable files].

#### **Phase1 Video**

## **Phase 2**

In this phase, you should extend code of phase 1 to build the full application and produce the final output file. Your application should support the different operation modes described in "Program Interface" section.

### **Phase 2 Deliverables:**

Each team is required to deliver the following:

A text file named **ID.txt** containing team members' names, IDs, and emails.

**Final project code** [Do not include executable files].

**Six comprehensive sample input files (test cases) and their output files. Sample input files must cover simple to complex scenarios.**

**Workload document.** For each part, which member implemented it.

## **Project Evaluation (More details will be added)**

These are the main points for project evaluation:

**Successful Compilation:** Your program must compile successfully with zero errors. Delivering the project with any compilation errors will make you lose a large percentage of your grade.

### **Object-Oriented Concepts:**

**Modularity:** A **modular** code does not mix several program features within the same unit (module). For example, the module that does the core of the simulation process should be separate from the module that reads the input file which, in turn, is separate from the module that implements the data structures. This can be achieved by:

Adding classes for each different entity in the system and each DS used.

Dividing the code in each class to several functions. Each function should be responsible for a single job. Avoid writing very long functions that do everything.

**Maintainability:** A maintainable code is the one whose modules are easily modified or extended without a severe effect on the other modules.

**Class responsibilities:** No class is performing the job of another class.

**Data Structures & Algorithms:** After all, this is what the course is about. You should be able to provide a concise description and a justification for: (1) the data structure(s) and algorithm(s) you used to solve the problem, (2) the **complexity** of the chosen algorithm,

and (3) the logic of the program flow.

**Interface modes:** Your program should support the three modes described in the document.

**Test Cases:** You should prepare different comprehensive test cases (at least 6). Your program should be able to simulate different scenarios not just trivial ones.

**Coding style:** How elegant and **consistent** is your coding style (indentation, naming convention, ...)? How useful and sufficient are your comments? This will be graded.

### **Notes:**

The code of any operation does NOT compensate for the absence of any other operation.

There is no bonus on anything other than the points mentioned in the bonus section.

### **Individuals Evaluation**

Each member must be responsible for writing some project modules (e.g. some classes or some functions) and must answer some questions showing that he/she understands both the program logic and the implementation details. The work load between team members must be almost equal.

The grade of **each student** will be divided as follows:

**[70%]** for his individual work (the project part he/she is responsible for).

**[25%]** for integrating his work with the work of ALL students who Finished or nearly finished their project part.

**[5%]** for cooperation with other team members and helping them if

they have any problems with their parts (helping does NOT mean to implement their parts for them).

If one or more members doesn't make their work, the other members will NOT be affected **as long as** the students who finished their part integrated all their parts together AND their part can be tested to see the output.

If a member couldn't finish his part, other members would integrate together to at least take the integration grade.

You should **inform the TAs** before the deadline **with a sufficient time (some weeks before)** if any problems happen between team members to be able to take grading action appropriately.

### **Bonus Criteria (Maximum 10% of Project Grade)**

**[2%] Night shifts:** Some trucks can work during company's off-hours

**[4%] Immediate Cargo Load/Reload:**

Ignore "Loading Rule" mentioned above. At each hour, if a cargo is waiting and a suitable truck (according to cargo assignment criteria) is available, start loading the cargo immediately.

This means that a truck can be loaded with different types of cargos at the same journey.

If a cargo is already loaded (but not moved yet) to a truck **not** of the same type as the cargo, then a truck of the same type becomes available, you should unload that cargo from the old truck and reload them to the new truck. Unload and reload times should be added to cargo waiting time.

**[2%] Truck Speed/Capacity:** Trucks of the same type may have different speeds and different capacities. The trucks of a specific type must be sorted according to speed and capacity (Develop a

priority equation based on these two factors). Higher priority trucks should be assigned first.

**[2%] Trucks Maintenance:** Trucks can sometimes need maintenance apart from their regular checkups. If this happens, they should be unavailable for some time. If the system needs this truck before its maintenance is over, this will cause the truck's speed to be decreased to its half till the end of the simulation.

Think about reasonable cases when a truck should get into maintenance.

**[4%] Delivery failure:** with a very low probability, a delivery might fail due to problems with its truck. In this case: (1) the truck should be moved to checkup, (2) the cargos are not delivered and need to be appended to the list of waiting cargos to be assigned later to other trucks.

## **Appendix A – Guidelines for Project Code**

The main classes of the project should be MarsStation, Cargo, Truck, Event, and UI (User Interface). In addition, you need lists of appropriate types to hold events, cargos, trucks,..etc.




### **Event Classes:**

There are three types of events: Preparation, Cancellation, and Promotion events. You should create a base class called "**Event**" that stores the event time and the related cargo info. This should be an abstract class with a pure virtual function "**Execute**". The logic of the Execute function should depend on the Event type.

For each type of the three events, there should be a class derived from **Event** class. Each derived class should store data related to its

operation. Also, each of them should override the function

**Execute** as follows:

1. PreparationEvent::Execute  should create a new cargo and add it to the appropriate list
2. CancelEvent::Execute  should cancel the requested normal cargo (if found and hasn't been loaded yet)
3. PromptEvent::Execute  should move a normal cargo to the VIP list and update the cargo's data (if found and hasn't been loaded yet)

### **Cargo and Truck Classes:**

Should have data members to store all info of cargos and trucks. In addition to appropriate member functions.

### **Company Class**

This class is the maestro class that manages the system.

It should have an appropriate ***list of Event pointers*** to store all events loaded from the file at system startup. Also it should have lists to hold cargos and trucks of different types and status.

It should have member functions to:

1. At program startup, open the input file and load trucks data to fill trucks list(s) and to load the events list.
2. On each hour,
  1. Execute the events that should be executed at that hour
  2. Check waiting cargo to assign them to available trucks
  3. Move cargos from waiting to loading to moving to delivered
  4. Move trucks from available to loading to moving to checkup to available again



5. Collect statistics that are needed to create output file
6. Calls UI class functions to print details on the output screen
  
3. Produce the output file at the end of simulation

## **UI Class**

This should be the class responsible for reading any inputs from the user and printing any information on the console. It contains the input and output functions that you should use to show status of the system at each hour.

At the end of each hour, this class should print the output screen (see Program Interface section) showing what has happened during that hour.

This is the only class whose functions can have input and output (cin and cout) lines

Input and output lines must NOT appear anywhere else in the program

CMP N102 / Spring 2022