# Assignment 1 – Fundamentals of Computer Programming CS-110

## Delivery Drone Simulator

| | |
|---|---|
| CLO 2 | Solve given real-world problem by applying appropriate programming concepts and techniques |
| CLO 4 | Perform effectively as an individual and as a member of a team. |

**Submitted By:**

| Full Name | CMS ID | Email |
|---|---|---|
| **Muhammad Mujtaba** | 540040 | mmujtaba.bese25seecs@seecs.edu.pk |
| **Azhan Ali** | 546256 | aazhan.bese25seecs@seecs.edu.pk |
| **Sheharyar Khalid** | 547279 | skhalid.bese25seecs@seecs.edu.pk |

**Submitted To:**

**Dr. Momina Moetesum**

School of Electrical Engineering and Computer Science (SEECS)

National University of Sciences & Technology (NUST)

**Class:** BESE 16B

**Batch:** 2k25

## Table of Contents

## Brief overview of the Requirements

Your program must:

- Start the Simulation
    - Display a welcome message and initial battery (e.g., 100%).
    - Ask the user to begin the delivery day.
- Environment Generation
    - Randomly generate:
        - Weather (1 = sunny, 2 = windy, 3 = rainy)
        - Obstacle (0 = none, 1 = yes)
        - Battery drain per trip (between 10–25%)
- Use **`rand()`** from **`<cstdlib>`** and **`srand(time(0))`** or variability.
- Decision Logic
    - If rainy → no flight, mark as delayed.
    - If windy and battery < 40% → return to base to recharge (+10%).
    - If obstacle detected → reroute and add extra battery drain (+5%).
    - Otherwise, perform delivery successfully.
- Optional Enhancements (for extra marks / curiosity)
    - Random “system malfunction” event (10% chance) → delivery failed.
    - User choice to recharge mid-mission.
    - Introduce a performance score (based on success – battery used).
    - Add delay timers using loops for realism.

## Overview / Problem Description

Aero Link is a C++ based console simulation of a drone which delivers tasks one after another.

- It detects obstacles and reroutes
- If weather is bad, it delays its flight to reduce damage
- It shows battery percentage and how much battery is used after each flight
- Gives a summary after 3 flights to the owner

## Execution Instructions

### Prerequisites

- CMake; Should be available in CLI
- C++ compiler linked with CMake; Should be available in CLI
- Visual Studio; For windows

### MacOS

1. Run `./setup.sh`
2. Follow instructions given in script
3. Run `./run.sh` in terminal
    a. The script will generate CMake files and run the file by itself automatically.

### Windows

1. Open the `src` folder in Visual Studio and run main.cpp

## Program Design/Logic

### List of Functions

- **`Weather randomWeather();`**
    - Returns random weather using rand(), used to simulate weather.
- **`Obstacle randomObstacle();`**
    - Returns random obstacle, this also uses rand()
- **`Condition randomMalfunction();`**
    - Has a random chance to return that the system has malfunctioned.
- **`bool checkWeather(int &battery, int &delay, int &recharges);`**
    - Checks the randomly chosen weather and prints messages for different weathers. If it is rainy, it increments delay and reports a delayed flight. If windy and battery < 40 the drone returns for recharge, otherwise it proceeds normally.
- **`bool checkObstacle(int &battery, int &reroutes);`**
    - Checks if their is a random obstacle or not, if yes, it reroutes and decreases more battery.
- **`bool checkMalfunction(int &failed);`**
    - Calls randomMalfunction() and displays message about malfunction if it really happened and increase time.

- **`void recharge(int &battery, std::string &choice, int &recharges);`**
    - Used to ask user if he wants to recharge the drone or not; if "yes" it increases battery by 30 (it can only reach at max, 100%) and increment variable used to represent the number of recharges. If "no" it continues without change; invalid input prints a warning but proceeds.
- **`void batteryDrain(int &battery, int &drain, int &failed);`**
    - Function subtracts drain from battery to simulate energy used for trip and if battery drops to 0, prints abort message, and increments failed. Amount of drain is random and is decided in void runSimulation(). It updates battery status.
- **`void preformance(int &battery);`**
    - Gives a score based on battery.
- **`void displayLocation(int &deliveries);`**
    - Prints message after the package is sent to different places
- **`bool startSimulation(int &deliveries, int &battery, int &start);`**
    - Shows the user a prompt, asking user to either enter 1 to begin or -1 to quit, simulation stops when we enter -1. Returns true only when user enters 1, allowing the simulation to proceed.
- **`void runSimulation(int &deliveries, int &battery, int &failed, int &drain, int &delay, int &recharges, int &reroutes);`**
    - Is used  to call all functions to get executed.
- **`void missionSummary(int &battery, int &deliveries, int &delay, int &failed, int &recharges, int &reroutes);`**
    - Prints the message showing how much battery is used, total deliveries, recharges and reroutes taken

## Control Flow

1. **`main`** seeds **`rand()`** using **`srand(time(0))`** and invokes **`runSimulation`**.
2. Each loop iteration calls **`startSimulation`** quitting aborts the mission early.
3. Environment updates evaluate weather, obstacle, and malfunction via the three **`check*`** helpers.
4. **`displayLocation`** and **`batteryDrain`** finalize successful deliveries.
5. **`recharge`** prompts for energy recovery before the next cycle.
6. A mission summary is generated through **`missionSummary`** in the main function.

### Environmental Randomness

1. Weather states are generated from Sunny, Windy, Rainy using rand().
2. Obstacles spawn with 50% probability rand().
3. Malfunctions occur with a 10% probability rand() % 10 == 0.

### Assumptions

1. Console input is limited to **1**, **–1**, **yes**, or **no**; other responses are given by user.
2. Three deliveries constitute one mission; additional locations are not queued automatically.

## Team Collaboration Summary

Our project/assignment was completed through active collaboration among all three team members. Each member contributed to the development process.

- Muhammad Mujtaba (Logic Designer)
    - Designed the logic and flowchart for the program.
    - Handled Git Repository.
- Azhan Ali (Mai n Programmer)
    - Implemented the environment generation system, including random weather, obstacle, and malfunction generation.
    - Made the entire program with multiple functions.
- Sheharyar Khalid (Tester/Documenter)
    - Thoroughly tested the program under multiple random scenarios to check if there is any problem with execution of logic.
    - Maintained the **`README.md`** documentation, organized all project information.

## AI Tool Reflection

- For Flowchart: **ChatGPT** to generate Figma
- Format inspiration for Output: **ChatGPT**
- For Function parameter documentation: **GitHub Copilot**

## Future improvements

1. We can use a **GUI** to visually show the movement of drone, its battery level, weather of the area around the drone and delivery status to the user of drone for better experience.
2. A more sophisticated approach for random number generated can be used other than **`rand`** such as modern C++ method using **`uniform_int_distribution`** and **`mt19937`**.

## GitHub Repo Link - Annex-I

https://github.com/aerolink-corp/aerolink-demo

## Code – Annex-II

### File Tree

```
src/
| --- main.cpp
| --- lib/
    | --- utils.cpp
    | --- utils.hpp
    | --- structures.hpp
```

## Code

**main.cpp**

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <chrono>
#include <thread>
#include <iomanip>
#include <string>

#include "utils.hpp"

// Main funtion which will execute
int main()
{
    // Initializing variables
    int delayedFlights = 0;
    int deliveredFlights = 0;
    int failedFlights = 0;
    int recharges = 0;
    int reroutes = 0;
    int drain;
    int battery = 100;
    // Displaying the welcome message
    std::cout <<
"==========================================================================================\n";
    std::cout << "                            WELCOME TO DRONE DELIVERY SIMULATOR\n";
    std::cout <<
"==========================================================================================\n\n";
    srand(time(0)); // Seeding the random generator
    // Calling the function which runs the simulation using loop
    runSimulation(deliveredFlights, battery, failedFlights, drain, delayedFlights, recharges, reroutes);

    // Calling the function to display the mission summary
    missionSummary(battery, deliveredFlights, delayedFlights, failedFlights, recharges, reroutes);

    // Displaying the end of simulation
    std::cout << "-----------------------------------------------------------------------------------------
-\n";
    std::cout <<
"\n==========================================================================================\n";
    std::cout << "                                       SIMULATION ENDED\n";
    std::cout <<
"==========================================================================================\n";

    return 0;
}
```

**utils.hpp**

```cpp
#pragma once
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <chrono>
#include <thread>
#include <iomanip>
#include <string>

#include "structures.hpp"

/// @brief Returns random weather using rand(), used to simulate weather.
/// @return A random Weather enum value.
Weather randomWeather();
/// @brief Returns random obstacle, this also uses rand()
/// @return A random Obstacle enum value.
Obstacle randomObstacle();
/// @brief Has a random chance to return that the system has malfunctioned.
/// @return A random Condition enum value.
Condition randomMalfunction();

/// @brief Checks the randomly chosen weather and prints messages for different weathers. If it is rainy,
it increments delay and reports a delayed flight. If windy and battery < 40 the drone returns for recharge,
otherwise it proceeds normally.
/// @param battery The current battery level of the drone.
/// @param delay The current delay time.
/// @param recharges The number of recharges.
/// @return True if the drone needs to return for recharge, false otherwise.
bool checkWeather(int &battery, int &delay, int &recharges);

/// @brief Checks if their is a random obstacle or not, if yes, it reroutes and decreases more battery.
/// @param battery The current battery level of the drone.
/// @param reroutes The number of reroutes.
/// @return True if the drone was rerouted, false otherwise.
bool checkObstacle(int &battery, int &reroutes);

/// @brief Calls _Condition randomMalfunction();_ and displays message about malfunction if it really
happened and increase time.
/// @param failed The number of failed attempts.
/// @return True if a malfunction occurred, false otherwise.
bool checkMalfunction(int &failed);

/// @brief Used to ask user if he wants to recharge the drone or not; if "yes" it increases battery by 30
(it can only reach at max, 100%) and increment variable used to represent the number of recharges. If "no"
it continues without change; invalid input prints a warning but proceeds.
/// @param battery The current battery level of the drone.
/// @param choice The user's choice regarding recharging.
/// @param recharges The number of recharges.
void recharge(int &battery, std::string &choice, int &recharges);

/// @brief Function subtracts drain from battery to simulate energy used for trip and if battery drops to
0, prints abort message, and increments failed. Amount of drain is random and is decided in `void
runSimulation()`. It updates battery status.
/// @param battery The current battery level of the drone.
/// @param drain The amount of battery to drain.
/// @param failed The number of failed attempts.
void batteryDrain(int &battery, int &drain, int &failed);

/// @brief Gives a score based on battery.
/// @param battery The current battery level of the drone.
void performance(int &battery);

/// @brief Prints message after the package is sent to different places
/// @param deliveries The number of deliveries made.
void displayLocation(int &deliveries);

/// @brief Shows the user a prompt, asking user to either enter 1 to begin or -1 to quit, simulation stops
when we enter -1. Returns true only when user enters 1, allowing the simulation to proceed.
/// @param deliveries The number of deliveries made.
/// @param battery The current battery level of the drone.
/// @param start The variable used to represent the start of the simulation.
/// @return True if the simulation should start, false otherwise.
bool startSimulation(int &deliveries, int &battery, int &start);
```

**utils.hpp**

```cpp
/// @brief Is used to call all functions to get executed.
/// @param deliveries The number of deliveries made.
/// @param battery The current battery level of the drone.
/// @param failed The number of failed attempts.
/// @param drain The amount of battery to drain.
/// @param delay The current delay time.
/// @param recharges The number of recharges.
/// @param reroutes The number of reroutes.
void runSimulation(int &deliveries, int &battery, int &failed, int &drain, int &delay, int &recharges, int
&reroutes);

/// @brief Prints the message showing how much battery is used, total deliveries, recharges and reroutes
taken
/// @param battery The current battery level of the drone.
/// @param deliveries The number of deliveries made.
/// @param delay The current delay time.
/// @param failed The number of failed attempts.
/// @param recharges The number of recharges.
/// @param reroutes The number of reroutes.
void missionSummary(int &battery, int &deliveries, int &delay, int &failed, int &recharges, int &reroutes);
```

**utils.cpp**

```cpp
#include "utils.hpp"
Weather randomWeather() { return static_cast<Weather>(rand() % 3 + 1); }
Obstacle randomObstacle() { return static_cast<Obstacle>(rand() % 2); }
Condition randomMalfunction()
{ return (rand() % 10 == 0) ? Condition::Malfunction : Condition::Normal; }
bool checkWeather(int &battery, int &delay, int &recharges)
{
    Weather weather = randomWeather(); // Calling function which returns random weather
    // if and else if statements which checks which weather is present
    if (weather == Weather::Rainy)
    {
        std::cout << "The flight is delayed because of the rainy weather!\n";
        std::this_thread::sleep_for(std::chrono::seconds(3));
        std::cout << "\nDrone taking off........\n";
        std::this_thread::sleep_for(std::chrono::seconds(1));
        std::cout << "The Drone has departed.\n";
        std::this_thread::sleep_for(std::chrono::seconds(3));
        delay++;
        return true;
    }
    else if (weather == Weather::Windy && battery < 40)
    {
        std::cout << "Weather is windy and battery is low, Returning to base for recharge\n";
        std::this_thread::sleep_for(std::chrono::seconds(2));
        std::cout << "\nDrone taking off........\n";
        std::this_thread::sleep_for(std::chrono::seconds(1));
        std::cout << "The Drone has departed.\n";
        std::this_thread::sleep_for(std::chrono::seconds(3));
        battery += 10;
        if (battery > 100)
            battery = 100;
        recharges++;
        return true;
    }
    else
    {
        std::cout << "\nDrone taking off........\n";
        std::this_thread::sleep_for(std::chrono::seconds(1));
        std::cout << "The Drone has departed.\n";
        std::this_thread::sleep_for(std::chrono::seconds(3));
    }
    return false;
}
```

**utils.cpp**

```cpp
bool checkObstacle(int &battery, int &reroutes)
{
    Obstacle obstacle = randomObstacle(); // Calling function to return random weather
    // if statement to check if the obstacle is present or not
    if (obstacle == Obstacle::Yes)
    {
        std::cout << "Obstacle detected! Rerouting...\n";
        std::this_thread::sleep_for(std::chrono::seconds(1));
        battery -= 5;
        reroutes++;
        return true;
    }
    return false;
}

bool checkMalfunction(int &failed)
{
    Condition malfunction = randomMalfunction(); // Calling function to return random malfunction
    // if statement to check if malfunction is generated or not
    if (malfunction == Condition::Malfunction)
    {
        std::cout << "System malfunction! Delivery failed.\n";
        failed++;
        std::this_thread::sleep_for(std::chrono::seconds(2));
        return true;
    }
    return false;
}

void recharge(int &battery, std::string &choice, int &recharges)
{
    // Asking user if he wants to recharge his battery before continuing to the delivery
    std::cout << "Your battery is " << battery << "%, " << "Do you want to recharge the drone before the
next delivery? (yes / no) : ";
    std::cin >> choice;
    if (choice == "yes")
    {
        std::cout << "Returning to the base for recharging....\n";
        std::this_thread::sleep_for(std::chrono::seconds(3));
        recharges++;
        battery += 30;
        if (battery > 100)
            battery = 100;
    }
    else if (choice == "no")
    {
        std::cout << "Continuing with the delivery....\n";
        std::this_thread::sleep_for(std::chrono::seconds(2));
    }
    else
    {
        std::cout << "Invalid Input!\n";
        std::cout << "Continuing....\n";
    }
}

void batteryDrain(int &battery, int &drain, int &failed)
{
    // Battery drain per trip
    battery -= drain;
    if (battery <= 0)
    {
        std::cout << "Battery Depleted! Aborting mission.\n";
        battery = 0;
        failed++;
    }
}

void performance(int &battery)
{
    int preformanceScore = battery;
    std::cout << std::setw(30) << "The performance score of the drone is: " << preformanceScore << '\n';
}
```

**utils.cpp**

```cpp
void displayLocation(int &deliveries)
{
    // switch statement to execute if the drone has completed it's delivery
    switch (deliveries + 1)
    {
    case 1:
        std::cout << "The Drone has successfully delivered to Location A.\n";
        break;
    case 2:
        std::cout << "The Drone has successfully delivered to Location B.\n";
        break;
    default:
        std::cout << "The Drone has successfully delivered to Location C.\n";
        break;
    }
}
bool startSimulation(int &deliveries, int &battery, int &start)
{
    std::cout << "-------------------------------------------------------------\n";
    std::cout << "Delivery #" << deliveries + 1 << " | Battery Level: " << battery << "%\n"; // Displaying
initial battery and delivery number
    std::cout << "-------------------------------------------------------------\n";
    std::cout << "Enter 1 to begin the delivery or -1 to quit: ";
    std::cin >> start;
    // if statement break the loop if the user enters 1
    if (start == -1)
    {
        std::cout << "Simulation Aborted!\n";
        return false;
    }
    else if (start != 1)
    {
        std::cout << "Invalid Input!\n";
        return false;
    }
    return true;
}
void runSimulation(int &deliveries, int &battery, int &failed, int &drain, int &delay, int &recharges, int
&reroutes)
{
    // While loop which will run 3 times
    while (deliveries < 3)
    {
        int start;
        bool proceed = startSimulation(deliveries, battery, start); // Calling the startSimulation funtion
to start the simulation
        std::string choice;
        drain = rand() % 16 + 10;

        if (!proceed)
        {
            if (start == -1) break;
            else continue;
        }
        checkWeather(battery, delay, recharges); // Calling the function to execute
        checkObstacle(battery, reroutes);        // Calling the function to execute

        bool malfunction = checkMalfunction(failed); // Calling the checkMalfunction function to start the
simulation
        if (malfunction) continue;

        // Switch statement to execute if the drone has completed it's delivery
        displayLocation(deliveries);

        // Calling the function to drain the battery after each delivery
        batteryDrain(battery, drain, failed);

        // if statement, so that after delivering to final location, it doesnt ask again to recharge
        if (deliveries < 2) recharge(battery, choice, recharges);
        // Incrementing deliveries at the end of the loop
        deliveries++;
    }
}
```

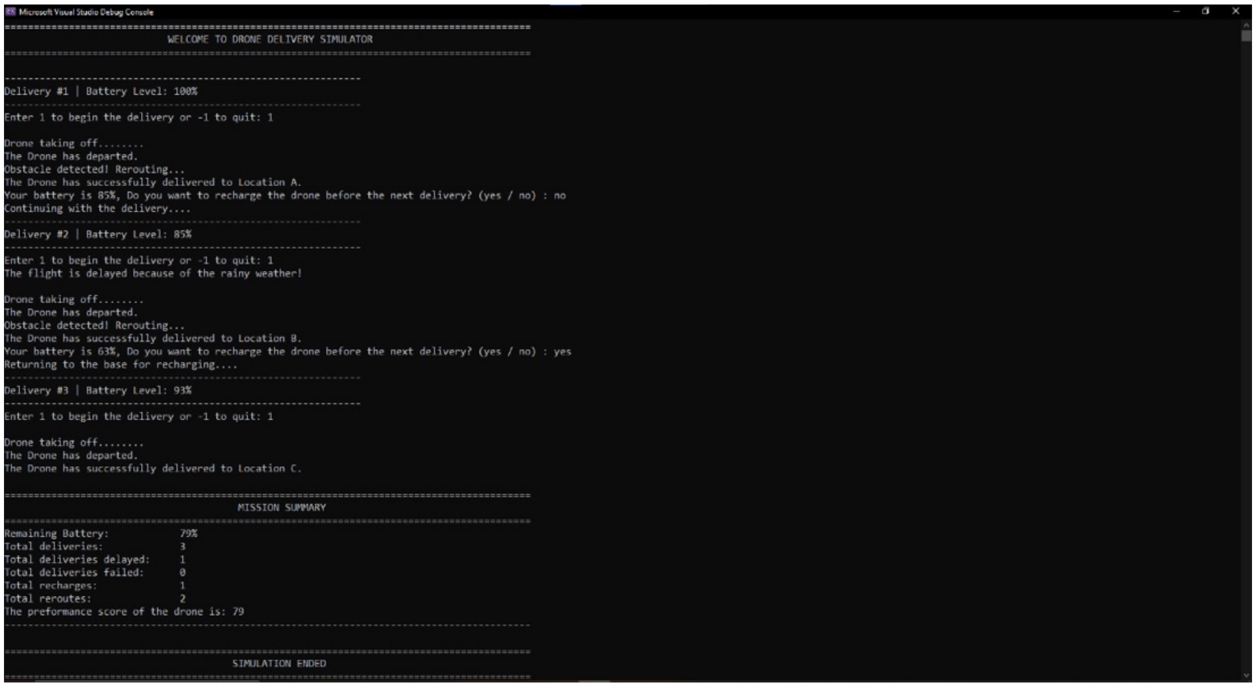
**utils.cpp**

```cpp
void missionSummary(int &battery, int &deliveries, int &delay, int &failed, int &recharges, int &reroutes)
{
    // Displaying the mission summary
    std::cout << 
"\n==========================================================================================\n";
    std::cout << "                                        MISSION SUMMARY\n";
    std::cout << 
"==========================================================================================\n";
    std::cout << std::left << std::setw(30) << "Remaining Battery: " << battery << "%\n";
    std::cout << std::setw(30) << "Total deliveries: " << deliveries << '\n';
    std::cout << std::setw(30) << "Total deliveries delayed: " << delay << '\n';
    std::cout << std::setw(30) << "Total deliveries failed: " << failed << '\n';
    std::cout << std::setw(30) << "Total recharges: " << recharges << '\n';
    std::cout << std::setw(30) << "Total reroutes: " << reroutes << '\n';
    performance(battery);
}
```

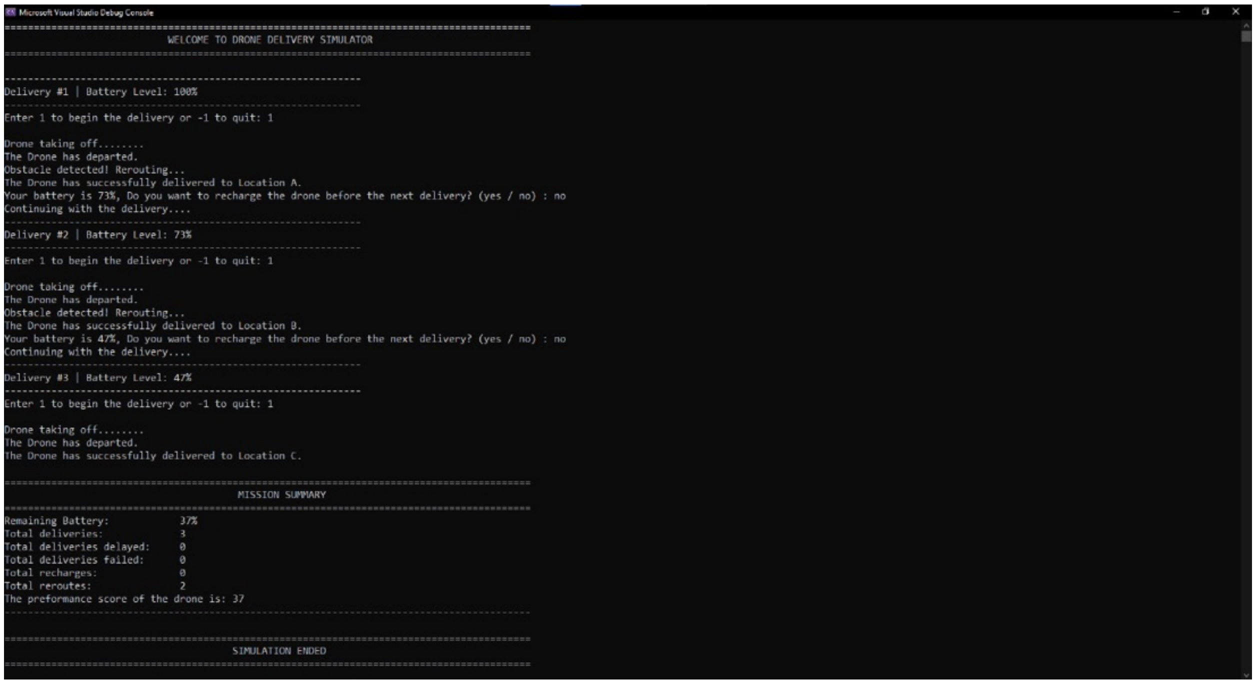
**structures.hpp**

```cpp
#pragma once

/// @brief Enum representing different weather conditions.
enum class Weather
{
    Sunny = 1,
    Windy = 2,
    Rainy = 3
};

/// @brief Enum representing the presence of an obstacle.
enum class Obstacle
{
    None = 0,
    Yes = 1
};

/// @brief Enum representing the condition of the drone.
enum class Condition
{
    Normal = 0,
    Malfunction = 1
};
```

## Output Screenshots

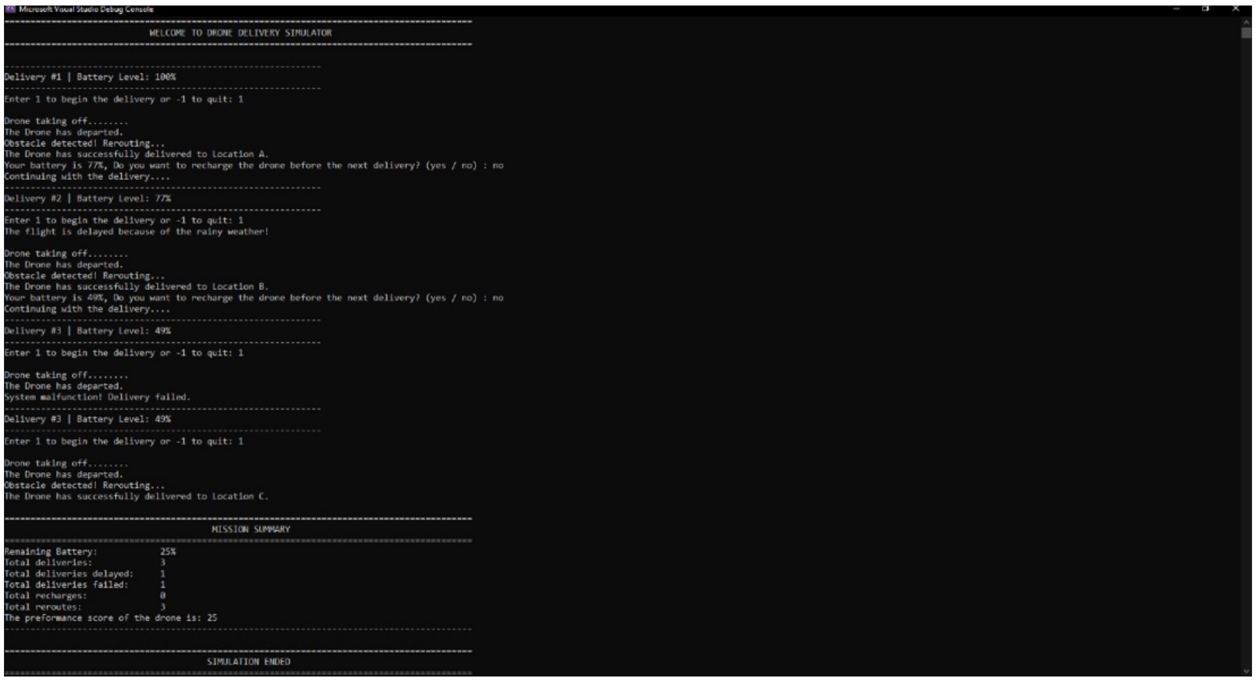


```
Microsoft Visual Studio Debug Console
==========================================================================================
                            WELCOME TO DRONE DELIVERY SIMULATOR
==========================================================================================

-------------------------------------------------------------
Delivery #1 | Battery Level: 100%
-------------------------------------------------------------
Enter 1 to begin the delivery or -1 to quit: 1

Drone taking off........
The Drone has departed.
Obstacle detected! Rerouting...
The Drone has successfully delivered to Location A.
Your battery is 85%, Do you want to recharge the drone before the next delivery? (yes / no) : no
Continuing with the delivery....
-------------------------------------------------------------
Delivery #2 | Battery Level: 85%
-------------------------------------------------------------
Enter 1 to begin the delivery or -1 to quit: 1
The flight is delayed because of the rainy weather!

Drone taking off........
The Drone has departed.
Obstacle detected! Rerouting...
The Drone has successfully delivered to Location B.
Your battery is 63%, Do you want to recharge the drone before the next delivery? (yes / no) : yes
Returning to the base for recharging....
-------------------------------------------------------------
Delivery #3 | Battery Level: 93%
-------------------------------------------------------------
Enter 1 to begin the delivery or -1 to quit: 1

Drone taking off........
The Drone has departed.
The Drone has successfully delivered to Location C.

==========================================================================================
                                        MISSION SUMMARY
==========================================================================================
Remaining Battery:            79%
Total deliveries:             3
Total deliveries delayed:     1
Total deliveries failed:      0
Total recharges:              1
Total reroutes:               2
The preformance score of the drone is: 79
------------------------------------------------------------------------------------------

==========================================================================================
                                        SIMULATION ENDED
==========================================================================================
```



```
Microsoft Visual Studio Debug Console
==========================================================================================
                            WELCOME TO DRONE DELIVERY SIMULATOR
==========================================================================================

-------------------------------------------------------------
Delivery #1 | Battery Level: 100%
-------------------------------------------------------------
Enter 1 to begin the delivery or -1 to quit: 1

Drone taking off........
The Drone has departed.
Obstacle detected! Rerouting...
The Drone has successfully delivered to Location A.
Your battery is 73%, Do you want to recharge the drone before the next delivery? (yes / no) : no
Continuing with the delivery....
-------------------------------------------------------------
Delivery #2 | Battery Level: 73%
-------------------------------------------------------------
Enter 1 to begin the delivery or -1 to quit: 1

Drone taking off........
The Drone has departed.
Obstacle detected! Rerouting...
The Drone has successfully delivered to Location B.
Your battery is 47%, Do you want to recharge the drone before the next delivery? (yes / no) : no
Continuing with the delivery....
-------------------------------------------------------------
Delivery #3 | Battery Level: 47%
-------------------------------------------------------------
Enter 1 to begin the delivery or -1 to quit: 1

Drone taking off........
The Drone has departed.
The Drone has successfully delivered to Location C.

==========================================================================================
                                        MISSION SUMMARY
==========================================================================================
Remaining Battery:            37%
Total deliveries:             3
Total deliveries delayed:     0
Total deliveries failed:      0
Total recharges:              0
Total reroutes:               2
The preformance score of the drone is: 37
------------------------------------------------------------------------------------------

==========================================================================================
                                        SIMULATION ENDED
==========================================================================================
```

https://github.com/aerolink-corp/aerolink-demo



Microsoft Visual Studio Debug Console

```
--------------------------------------------------------------------------------------
                           WELCOME TO DRONE DELIVERY SIMULATOR
--------------------------------------------------------------------------------------


------------------------------------------------------------
Delivery #1 | Battery Level: 100%
------------------------------------------------------------
Enter 1 to begin the delivery or -1 to quit: 1

Drone taking off........
The Drone has departed.
Obstacle detected! Rerouting...
The Drone has successfully delivered to Location A.
Your battery is 77%, Do you want to recharge the drone before the next delivery? (yes / no) : no
Continuing with the delivery....
------------------------------------------------------------
Delivery #2 | Battery Level: 77%
------------------------------------------------------------
Enter 1 to begin the delivery or -1 to quit: 1
The flight is delayed because of the rainy weather!

Drone taking off........
The Drone has departed.
Obstacle detected! Rerouting...
The Drone has successfully delivered to Location B.
Your battery is 49%, Do you want to recharge the drone before the next delivery? (yes / no) : no
Continuing with the delivery....
------------------------------------------------------------
Delivery #3 | Battery Level: 49%
------------------------------------------------------------
Enter 1 to begin the delivery or -1 to quit: 1

Drone taking off........
The Drone has departed.
System malfunction! Delivery failed.
------------------------------------------------------------
Delivery #3 | Battery Level: 49%
------------------------------------------------------------
Enter 1 to begin the delivery or -1 to quit: 1

Drone taking off........
The Drone has departed.
Obstacle detected! Rerouting...
The Drone has successfully delivered to Location C.

======================================================================================
                                    MISSION SUMMARY
======================================================================================
Remaining Battery:            25%
Total deliveries:             3
Total deliveries delayed:     1
Total deliveries failed:      1
Total recharges:              0
Total reroutes:               3
The preformance score of the drone is: 25
--------------------------------------------------------------------------------------

======================================================================================
                                   SIMULATION ENDED
======================================================================================
```

https://github.com/aerolink-corp/aerolink-demo