This is a sample program based on my logic.

**CODE:**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <string>

using namespace std;

//prototyping

void start\_Day(int& battery); //no return so use void

int getWeather();

bool checkObstacle();

bool checkLoad();

string get\_location(int delivery\_number);

void deliverPackage(string location, int& battery, int& successful, int& failed, int& delayed);

void displaySummary(int successful, int failed, int delayed, int battery);

int main() {

srand(time(0)); //for generating different values

int battery = 100;

int successful = 0;

int failed = 0;

int delayed = 0;

start\_Day(battery);

for (int i = 1; i <= 3; i++) {

string currentLocation = get\_location(i); // Get the location name

cout << "\nDelivery " << i << ":" <<endl;

deliverPackage(currentLocation, battery, successful, failed, delayed);

}

displaySummary(successful, failed, delayed, battery);

return 0;

}

//defining functions.

void start\_Day(int& battery) { //display for the start of the delivery day.

cout << ".....Welcome to Delivery Drone System ....."<<endl;

cout << "Initial Battery level: " << battery << "%"<<endl;

cout << "Starting the deliveries."<<endl;

}

int getWeather() {

int weather = (rand() % 3) + 1; //generates random weather.

if (weather == 1) {

cout << "The Weather is sunny."<<endl;

}

else if (weather == 2) {

cout << "The Weather is windy."<<endl;

}

else {

cout << "The Weather is rainy."<<endl;

}

return weather;

}

bool checkObstacle() {

int obstacle = rand() % 2; // Generates 0 or 1: 0 for no obstacle, 1 for obstacle.

if (obstacle) {

cout << "Oops, there are obstacles on the way." << endl;

return true;

}

else {

cout << "No obstacles found."<<endl;

return false;

}

}

bool checkLoad() { //for the restriction on load.

int load = rand() % 2;

if (load) {

cout << "The load is heavy, This will affect its efficiency."<<endl;

return true;

}

else {

cout << "Load is fine for the drone."<<endl;

return false;

}

}

void deliverPackage(string location, int& battery, int& successful, int& failed, int& delayed) {

cout << "Delivery to: " << location << endl;

int weather = getWeather();

bool obstacle = checkObstacle();

bool heavyLoad = checkLoad();

int drain = (rand() % 16) + 10;

bool weatherDelay = false; //to fix the number of delayed deliveries.

if (heavyLoad) {

drain += 15;

}

if (weather == 3) {

cout << "Deliveries delayed due to rain."<<endl;

weatherDelay = true;

}

else if (weather == 2 && battery < 40) {

cout << "Cannot proceed in windy weather with this battery."<<endl;

battery += 15;

weatherDelay = true;

}

if (weatherDelay) {

delayed++;

}

else if (obstacle) {

cout << "Changing the route. Extra battery usage."<<endl;

battery -= drain + 8;

successful++;

}

else {

cout << "Package delivered successfully!"<<endl;

battery -= drain;

successful++;

}

if (battery <= 0) {

cout << "Battery level is ZERO! Drone cannot perform."<<endl;

if (successful > 0 && !weatherDelay) {

successful--;

}

failed++;

battery = 0;

}

else if (battery > 100) {

battery = 100;

}

cout << "Remaining Battery Level: " << battery << "%"<<endl;

}

void displaySummary(int successful, int failed, int delayed, int battery) {

cout << "\n........... DELIVERIES SUMMARY..............."<<endl;

cout << "Number of successful deliveries: " << successful << endl;

cout << "Number of failed deliveries: " << failed << endl;

cout << "Number of delayed deliveries: " << delayed << endl;

cout << "Battery Level Remaining: " << battery << "%"<<endl;

}

string get\_location(int delivery\_number) {

string location;

if (delivery\_number == 1) {

location = "Location A";

}

else if (delivery\_number == 2) {

location = "Location B";

}

else if (delivery\_number == 3) {

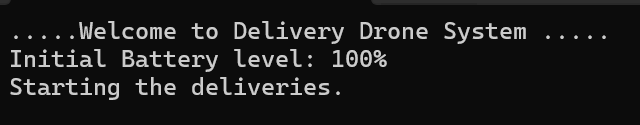
location = "Location C";

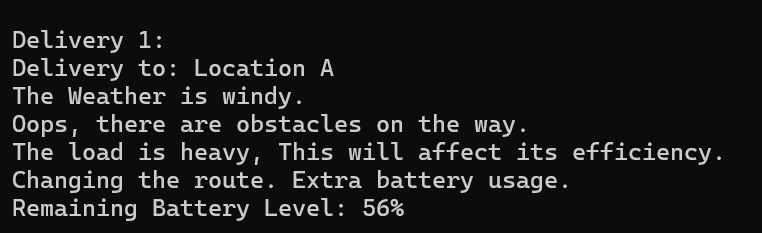
}

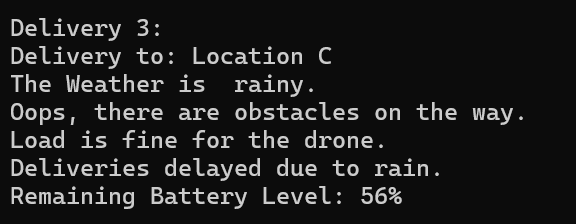
return location;

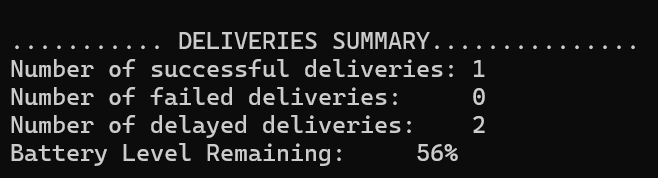
}

**OUTPUT TESTS:**









**AI HELP:**

1) To understand the logic and placement of counters.

2) To correct my *delayed counter* by adding a bool flag *weatherDelay.*

3) Understand the calling of functions.