Background:

You have been tasked with creating a simulation of an airport under construction to test how the airport's 2 runways will deal with a high load of arriving and departing aircraft. Aircraft need to wait a minimum period of time when landing or taking off behind another aircraft to avoid collisions or dangerous flying conditions. When an aircraft needs to wait to land it will circle in the air. If the aircraft has circled 3 times it will divert to another airport. Different types of aircraft will have a different time for which they occupy the runway when landing and taking off.

Task Description:

1. Types needed:

You will have an enumerator of the different Aircraft Types – use the following code:

```
type TAircraft is (A320, A330, A340, A350, A380);
for TAircraft use (1, 2, 3, 4, 5);
```

The top line refers to the type of aircraft, similar to the midterm assignment. The bottom line is the values for each type of aircraft and will be used when implementing the delays for landing, take-off, refueling, disembarking and boarding.

To get the value of the TAircraft you can use: **EnumType'Enum_Rep(Enum)**

Furthermore: **Create an exception called Diversion.**

2. Printer:

Create a protected object which will be responsible for all printing to the console throughout the program.

3. Random Aircraft Generator:

Create a protected object which will be responsible for generating random aircraft types from the TAircraft enumerator. The protected object has 2 entries:

- Init – reset the random generator.
- Generate function – returns random TAircraft

4. Airport

Create a protected object called **Airport** which has 2 private variables:

- Number of Aircraft at the Terminal – Type Natural – initially 0.
- Maximum number of aircraft – Type Natural – initially 0.

The protected has 3 entities within the public part:

- Procedure Init which takes a Natural Number as parameter and sets the maximum number of aircraft.
- Entry Arrival – accepts the entry when the number of aircraft at the terminal is less than the maximum.
    - Increases the number of aircraft at the terminal by 1.
    - Prints the number of aircraft at the terminal.
- Entry Departure – accepts the entry when the number of aircraft at the terminal is greater than 0.

    - Decreases the number of aircraft at the terminal by 1.
    - Prints the number of aircraft at the terminal.

5. Fuel

Create a regular task called **Fuel** which is responsible for refueling the aircraft. It has 1 entry:

- The entry is called Refuel and has 2 parameters:
    - Natural Number which is the flight number.
    - Aircraft Type
- The entry prints the flight number, and that the aircraft is being refueled.
- It then delays by a duration which is the value of the TAircraft aircraft type.

The task should immediately be able to accept multiple requests to refuel and if the task has not been called in 30 seconds, then it should terminate (exit).

6. Runway

Create a task type called **Runway** which takes in a **String Pointer** as the name of the runway. The Runway task has 2 entries:

- Entry Takeoff which has 2 parameters (Natural number which is the flight number and Aircraft Type).
    - Calls the departure entry of the Airport.
    - Prints the flight number and that the aircraft is taking off and the name of the runway from which it is taking off.
    - Delays by a duration which is: Value of TAircraft / 5.0.
- Entry Land which has 2 parameters (Natural number which is the flight number and Aircraft Type).
    - Calls the arrival entry of the Airport.
    - Prints the flight number and that the aircraft landing and the name of the runway from which it is taking off.
    - Delays by a duration of 0.5.

The task can only accept 1 entry at a time and if it has not been called in 30 seconds, it should terminate (exit).

Create Runway R1 with the name "12R/30L".
Create Runway R2 with the name "12L/30R".
(If you are interested to know how runway naming works:
https://en.wikipedia.org/wiki/Runway#Naming – not needed for understanding of the task).

7. Flight

Create a task type **Flight** which has 1 entry:

- Entry Init – 2 parameters: Num (Natural number), A (Aircraft Type)
    - Sets the Flight number and the aircraft type of the Flight.
- Accepts this entry first before anything else.

Once the Flight has been initialized, the flight tries to land by calling the Runway.

- Aircraft land using R1.
- If the runway is clear and landing can be initiated, print the flight number and aircraft type and that it is landing.

- If landing cannot be initiated because the runway is occupied, delay by 0.2 and print that the flight is circling.
- Try to land again.
- If the flight has circled 3 or more times, raise the Diversion exception.

Once the aircraft has landed, it needs to be refueled.
After refueling the aircraft needs to wait on the ground for boarding to take place. This duration is the value of (Aircraft type * 2.0)

Once complete the aircraft will try to takeoff. The aircraft will take off from R2.

- If the runway is clear and takeoff can be initiated, print the flight number and aircraft type and that it is taking off.
- If takeoff cannot be initiated because the runway is occupied, delay by 0.2 and print that the flight is waiting for takeoff.

Handle the Diversion exception at the end of the task by printing a message which contains the flight number, aircraft type and that the flight has circled too much and needs to divert.

Handle any other tasking exceptions.

Create an array of 10 flights.

8. Main

In the main body initialize the random generator.

Initialize the Airport with a maximum size of 10 aircraft at the terminal.

Loop through the flights array and initialize each flight with a flight number (I) and a random aircraft type.

delay 0.5 between the initialization of each flight.

The application should terminate – will take 30 seconds from the last takeoff before it terminates.

Make sure all printing messages are detailed and says from which task it is printing.

Depending on the random generation it is possible that flights will be diverted.