AERSP 424: Advanced Computer Programming

Homework 1 Spring 2024

Submission Instructions:

- Submit a .zip file containing your .cpp and/or .h files.
- Bonus points (5 points) if you use Git, has a history of commits, and submit a URL to your repository.
- Your code needs to be compilable and contains some comments.
- Only the final submission will be graded.
- The submission deadline is at 11:59 PM on Friday 2/23/2023 (The late policy mentioned in the syllabus will be applied).
- Explicitly declare variables with a proper name (easy to understand) and datatype (reflect the real-world scenario *if possible*).

Question 1 (15 points):

Assuming you are about to fly a plane that carries that following items: (1) its own empty weight, (2) baggage, (3) front seat passengers, (4) rear seat passengers, and (5) usable fuel. Each item contributes to the gross weight and the C.G. location of the airplane. Write a program to receive information below from the user and store them in appropriate variables.

- Airplane empty weight (pounds)
- Airplane empty-weight moment (pounds-inches)
- The number of front seat occupants
- Weight of each front seat occupant (pounds)
- Front seat moment arm (inches)
- The number of rear seat occupants
- Weight of each rear seat occupant (pounds)
- Rear seat moment arm (inches)
- The number of gallons of usable fuel (gallons)
- Usable fuel weights per gallon (pounds)
- Fuel tank moment arm (inches)

- Baggage weight (pounds)
- Baggage moment arm (inches)

Then, check if the gross weight and the C.G. location are within the airplane design limits using the moment equation.

$$Moment = distance \times force$$

If not, calculate the amount of fuel that need to be added or drained in order to meet the limits. Also, print out the new value of the gross weight and the C.G. location.

The airplane design limits are as follow:

- Maximum allowable gross weight: 2950 lbs.
- Forward C.G. limit: 82.1 inches & Aft C.G. limit: 84.7 inches
- Use a decimal precision of 0.01 lbs. when adding or draining fuel.

Note: All calculations must be done by a computer, e.g., no hard-coded number other than the ones provided. You may use the following numbers to check your result (but the code needs to be able to receive user inputs).

- Airplane empty weight (pounds): 2050
- Airplane empty-weight moment (pounds-inches): 155400
- The number of front seat occupants: 2
- Weight of each front seat occupant (pounds): 180, 150
- Front seat moment arm (inches): 85
- The number of rear seat occupants: 2
- Weight of each rear seat occupant (pounds): 160, 170
- Rear seat moment arm (inches): 121
- The number of gallons of usable fuel (gallons): 44
- Usable fuel weights per gallon (pounds): 6
- Fuel tank moment arm (inches): 75
- Baggage weight (pounds): 10
- Baggage moment arm (inches): 140

Question 2 (10 points):

Given a picture of a map around the University Park airport, design a C++ container that stores information shown in the picture, specifically:

- There are 160-mile flights between "SCE" and "PHL".
- There are 640-mile flights between "SCE" and "ORD".
- There are 220-mile flights between "SCE" and "EWR".

Hint: The designed container can be a single container, a container of container(s), multiple containers, or a combination of all the above.

Question 3 (10 points):

Create a class named "*Plane*" with the following class members.

Private members:

- Double variables named "*pos*", "*vel*", and "*distance*".
- A boolean variable named "*at_SCE*".
- String variables named "*origin*" and "*destination*".
- A container from Question 2.

Public members:

- A constructor that takes in two strings "*from*" and "*to*" as input.

- A deconstructor.
- A function named "*operate*" with a double variables "*dt*" as an input, and return nothing.
- get functions for "*pos*", "*origin*", "*destination*", and "*at_SCE*" variables and get/set functions for "*vel*" variable.

Question 4 (15 points):

Implement functions in a class from Question 3 as follows.

All get/set functions return or set the value of their corresponding variable.

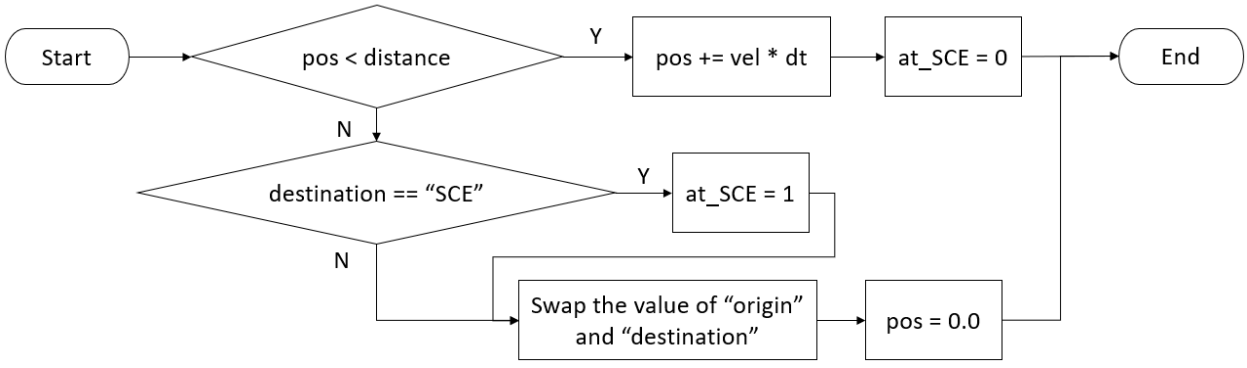
A constructor

- Store the input string "*from*" in "*origin*" variable.
- Store the input string "*to*" in "*destination*" variable.
- Set the value of "*distance*" to the distance between the origin and the destination airports (Hint: use your container from Question 2).
- Initialize the value of "*pos*", "*vel*", and "*at_SCE*" variables to zero.
- Print out a string "*Plane Created at* " followed by the memory address where this plane object is stored.

A deconstructor

- Print out a string "*Plane Destroyed*".

An "*operate*" function contains the following flowchart.



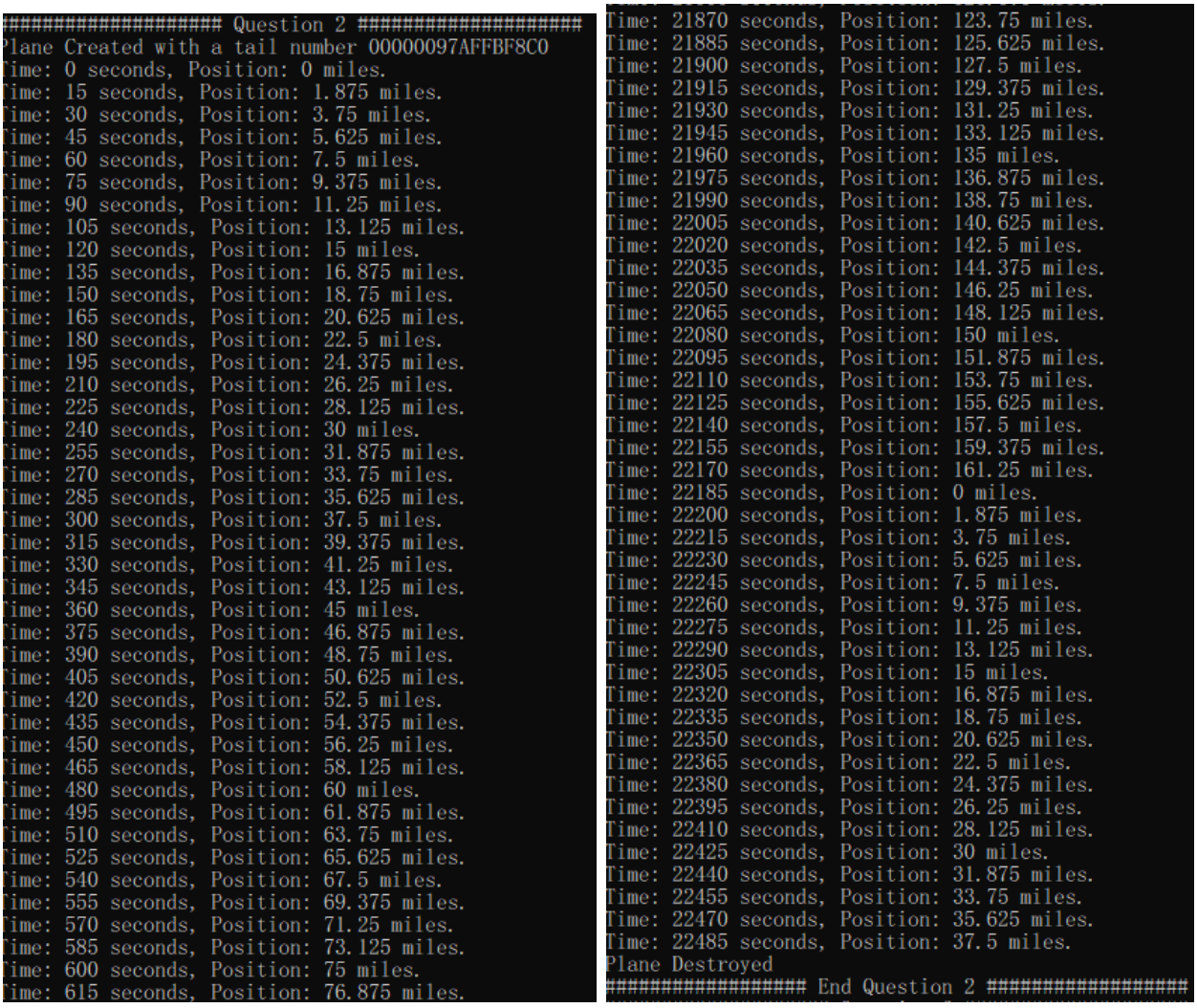
Note: If time-related variables go under zero, set them to zero. All time-related variables are in seconds.

Question 5 (15 points):

Pick a flight speed between 400-500 mph. In your main function, instantiate an object from the "*Plane*" class written in Question 3 & 4.

This object represents an airplane of the chosen pair of arrival & departure flights. Use the set function for "*vel*" to set the speed of the airplane. Then, pick a timestep between [10, 100], choose the maximum number of iterations between [1000, 2000], and create an iterative statement that call the "*operate*" function member for each airplane object with timestep being an input. Print out the airplane position at each timestep until the maximum number of iterations is reached.

Example printed outputs are as follows.



Question 6 (5 points):

Create a class "*Pilot*" with the following members and implementation.

Private members:

- A string variable named "*name*".

Public members:

- A constructor that prints out the pilot's name, memory address, and say that the pilot is at the gate, ready to board the plane.
- A deconstructor that prints out the pilot's name and say that the pilot is out of the plane.
- A get function of the variable "*name*".
- A pointer, named "*myPlane*", to an object of the "*Plane*" class.

Question 7 (20 points):

Now, imagine that there are two pilots flying a plane, e.g., the Pilot-in-Command and the Co-Pilot. Each of them takes turns to control the plane whenever the plane is landed at SCE. For example, at the beginning, the first pilot takes off from SCE, flies to the destination, files back, and lands at SCE. After that, the second pilot takes control of the plane, then takes off from SCE, flies to the destination, files back, and lands at SCE. After that, the first pilot takes over, and this process keeps repeating. Write a code to simulate this scenario.

Use the same velocity, timestep, and the maximum number of iterations as Question 5. Before starting an iteration prints out the pilots' name, pilots' memory address, and the memory address of the plane that they are controlling. When the plane lands at SCE, print out the memory address of the plane and say that it is at SCE. After switching who's in control of the plane, prints out the pilots' name and the memory address of the plane that they are controlling. Pilot objects are instantiated from the "*Pilot*" class in Question 6. The plane object is from Question 3 & 4.

The example printed output is shown below.

```
#################### Question 6 ####################
Pilot Alpha with certificate number 000001EBB1770690 is at the gate, ready to board the plane.
Pilot Bravo with certificate number 000001EBB17704D0 is at the gate, ready to board the plane.
Plane Created with a tail number 000001EBB1766BC0
Pilot Alpha with certificate number 000001EBB1770690 is in control of a plane: 000001EBB1766BC0
Pilot Bravo with certificate number 000001EBB17704D0 is in control of a plane: 0000000000000000

The plane 000001EBB1766BC0 is at SCE
Pilot Bravo with certificate number 000001EBB17704D0 is in control of a plane: 000001EBB1766BC0
Pilot Alpha with certificate number 000001EBB1770690 is in control of a plane: 0000000000000000

The plane 000001EBB1766BC0 is at SCE
Pilot Alpha with certificate number 000001EBB1770690 is in control of a plane: 000001EBB1766BC0
Pilot Bravo with certificate number 000001EBB17704D0 is in control of a plane: 0000000000000000

The plane 000001EBB1766BC0 is at SCE
Pilot Bravo with certificate number 000001EBB17704D0 is in control of a plane: 000001EBB1766BC0
Pilot Alpha with certificate number 000001EBB1770690 is in control of a plane: 0000000000000000

The plane 000001EBB1766BC0 is at SCE
Pilot Alpha with certificate number 000001EBB1770690 is in control of a plane: 000001EBB1766BC0
Pilot Bravo with certificate number 000001EBB17704D0 is in control of a plane: 0000000000000000

The plane 000001EBB1766BC0 is at SCE
Pilot Bravo with certificate number 000001EBB17704D0 is in control of a plane: 000001EBB1766BC0
Pilot Alpha with certificate number 000001EBB1770690 is in control of a plane: 0000000000000000

The plane 000001EBB1766BC0 is at SCE
Pilot Alpha with certificate number 000001EBB1770690 is in control of a plane: 000001EBB1766BC0
Pilot Bravo with certificate number 000001EBB17704D0 is in control of a plane: 0000000000000000

The plane 000001EBB1766BC0 is at SCE
Pilot Bravo with certificate number 000001EBB17704D0 is in control of a plane: 000001EBB1766BC0
Pilot Alpha with certificate number 000001EBB1770690 is in control of a plane: 0000000000000000

The plane 000001EBB1766BC0 is at SCE
Pilot Alpha with certificate number 000001EBB1770690 is in control of a plane: 000001EBB1766BC0
Pilot Bravo with certificate number 000001EBB17704D0 is in control of a plane: 0000000000000000

Pilot Bravo is out of the plane.
Pilot Alpha is out of the plane.
Plane Destroyed
################## End Question 6 ##################
```

Question 8 (20 points):

If you write a code in Question 7 using an old C/CPP-style pointer, rewrite it using modern CPP-style pointer. If you write a code in Question 7 using a modern CPP-style pointer, rewrite it using an old C/CPP-style pointer.

Note: In the modern CPP-style, there will NOT be any '*', '*new*', '*delete*', '*malloc*', or '*free*' keywords in the code.