CS 162: Computer Science II

Algorithm Design Document

Make a copy before you begin (File -> Make a copy). Add the Assignment # above and complete the sections below BEFORE you begin to code. The sections will expand as you type. When you are finished, download this document as a PDF (File -> Download -> PDF) and submit to D2L.

This document contains an interactive checklist. To mark an item as complete, click on the box (the entire list will be highlighted), then right click (the clicked box will only be highlighted), and choose the checkmark.

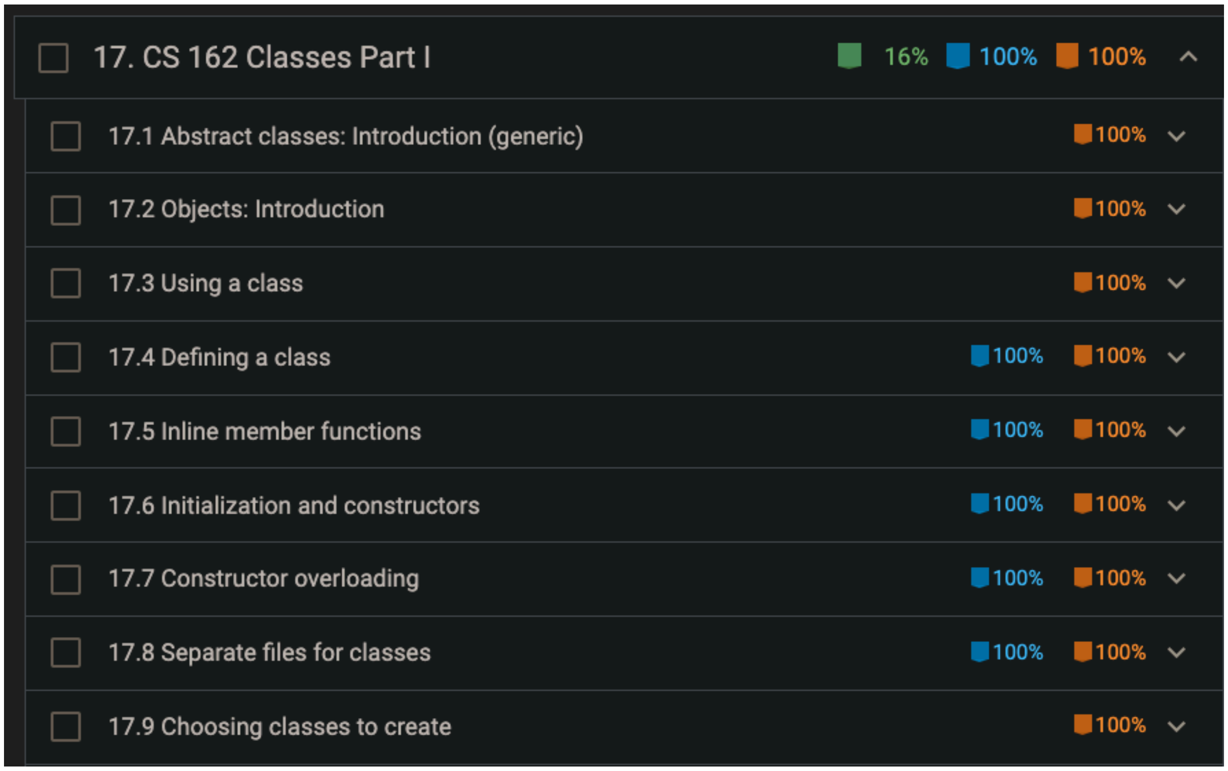
Planning your program before you start coding is part of the development process. In this document you will:

- ✓ ~~Paste a screenshot of your zyBooks Challenge and Participation %~~
- ✓ ~~Paste a screenshot of your assigned zyLabs completion~~
- ✓ ~~Write a detailed description of your program, at least two complete sentences~~
- ✓ ~~If applicable, design a sample run with test input and output~~
- ✓ ~~Identify the program inputs and their data types~~
- ✓ ~~Identify the program outputs and their data types~~
- ✓ ~~Identify any calculations or formulas needed~~
- ✓ ~~Write the algorithmic steps as pseudocode~~

1. zyBooks

Add your zyBooks screenshots for the % and assigned zyLabs completions below. Required percentages: all **assigned** zyLabs, Challenge Activity with at least 70%, and Participation Activity with at least 80%.

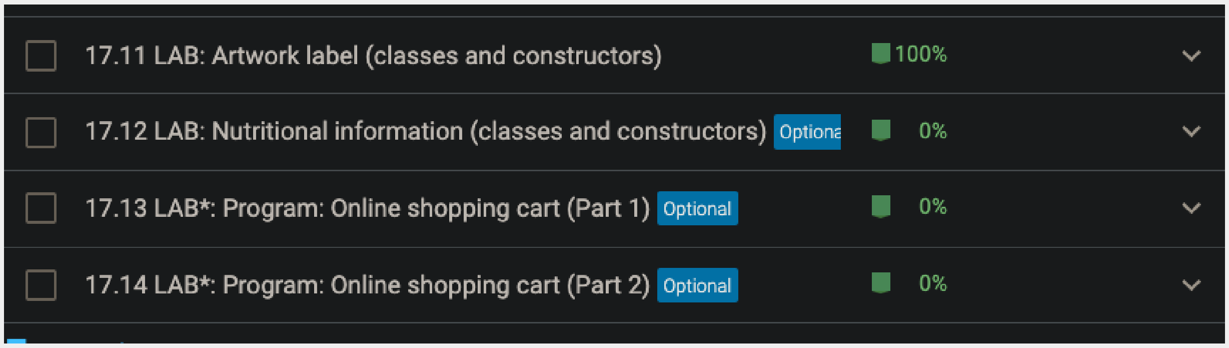
Challenge and Participation % screenshot:



17. CS 162 Classes Part I 🟩 16% 🟦 100% 🟧 100%

17.1 Abstract classes: Introduction (generic)		🟧 100%
17.2 Objects: Introduction		🟧 100%
17.3 Using a class		🟧 100%
17.4 Defining a class	🟦 100%	🟧 100%
17.5 Inline member functions	🟦 100%	🟧 100%
17.6 Initialization and constructors	🟦 100%	🟧 100%
17.7 Constructor overloading	🟦 100%	🟧 100%
17.8 Separate files for classes	🟦 100%	🟧 100%
17.9 Choosing classes to create		🟧 100%

Assigned zyLabs completion screenshot:



17.11 LAB: Artwork label (classes and constructors)	🟩 100%
17.12 LAB: Nutritional information (classes and constructors) Optiona	🟩 0%
17.13 LAB*: Program: Online shopping cart (Part 1) Optional	🟩 0%
17.14 LAB*: Program: Online shopping cart (Part 2) Optional	🟩 0%

2. Program Description

In the box below, describe the purpose of the program. You must include a detailed description with at least two complete sentences.

Program description:

```
This program uses nested classes, Airplane and Fleet, to set private and
public member functions and data. The program is multi-file, and allows a
user to add a plane, remove a plane, view the planes, search planes, etc
using constructors, functions, etc.
```

3. Sample Run

If you are designing your own program, you will start with a sample run. **Imagine** a user is running your program - what will they see? What inputs do you expect, and what will be the outputs from the given inputs? Choose test data you will use to test your program. Calculate and show the expected outputs. Use the sample run to test your program.
Do not simply copy the sample run from the assignment instructions!

Sample run:

```
Welcome to the airplane collection program!

What is the name of the airplane collection file? notAFile.txt

*** The file notAFile.txt did not open. Type 'Q' to quit, or try again now: airplanes.txt

360 plane data was successfully inserted.
Skyhawk 172 plane data was successfully inserted.
K35 Bonanza plane data was successfully inserted.
RangeMaster H plane data was successfully inserted.
Tomahawk plane data was successfully inserted.
M20R Ovation plane data was successfully inserted.
C23 Sundowner plane data was successfully inserted.
RV-12 plane data was successfully inserted.
TB-21 GT Trinidad plane data was successfully inserted.
RV-9 plane data was successfully inserted.
152 plane data was successfully inserted.
Tiger plane data was successfully inserted.
Super Cub plane data was successfully inserted.


Choose an option:
L: List All Planes
M: List Planes by Make
A: Add a New Plane
R: Remove a Plane
Q: Save and Quit

A
```

```
What is the model (name) of the airplane? Malibu Mirage
What is the make (manufacturer) of the airplane? Piper
What is the fuel capacity in gallons? One hundred and twenty
Please enter a decimal number for fuel capacity between 1.00 and 150.00: 120.00
What is the empty weight? 20500
The weight must be a whole number between 1 and 3000 pounds: 2435
What is the horsepower of the engine? 550
The horsepower must be a whole number between 1 and 400: 350
What is the range? 1342
What is the cruise speed? 212
Malibu Mirage plane data was successfully inserted.

Choose an option:
L: List All Planes
M: List Planes by Make
A: Add a New Plane
R: Remove a Plane
Q: Save and Quit

L

Model              Make           Fuel Capacity  Empty Weight   Horsepower     Range          Cruise speed
-----------------------------------------------------------------------------------------------------
1. 152                 Cessna         26             1081           110            414            106
2. 360                 Lancair        43             1090           180            990            208
3. C23 Sundowner       Beechcraft     57             1494           180            564            115
4. K35 Bonanza         Beechcraft     70             1832           250            534            168
5. M20R Ovation        Mooney         89             2205           280            969            189
6. Malibu Mirage       Piper          120            2435           350            1342           212
7. RV-12               Vans Aircraft  20             750            100            451            119
8. RV-9                Vans Aircraft  36             1057           160            616            163
9. RangeMaster H       Navion         40             1945           330            1381           160
10. Skyhawk 172         Cessna         53             1663           180            515            123
11. Super Cub           Piper          36             845            125            449            96
12. TB-21 GT Trinidad   Socata         88             1911           250            1025           168
13. Tiger               Grumman        51             1360           180            529            139
14. Tomahawk            Piper          30             1128           112            383            107

Choose an option:
L: List All Planes
M: List Planes by Make
A: Add a New Plane
R: Remove a Plane
Q: Save and Quit

M

Please type the make of the airplanes you would like to list: Jabiru
```

```
Model              Make           Fuel Capacity  Empty Weight   Horsepower     Range
Cruise speed
-------------------------------------------------------------------------------------------------
----

Choose an option:
L: List All Planes
M: List Planes by Make
A: Add a New Plane
R: Remove a Plane
Q: Save and Quit

M

Please type the make of the airplanes you would like to list: Piper

Model              Make           Fuel Capacity  Empty Weight   Horsepower     Range          Cruise speed
------------------------------------------------------------------------------------------------------
6. Malibu Mirage       Piper          120            2435           350            1342           212
11. Super Cub           Piper          36             845            125            449            96
14. Tomahawk            Piper          30             1128           112            383            107

Choose an option:
L: List All Planes
M: List Planes by Make
A: Add a New Plane
R: Remove a Plane
Q: Save and Quit

R
Which index would you like to remove (1 – 14)? 0
Invalid Index. Please type an index between 1 and 14: 15
Invalid Index. Please type an index between 1 and 14: four
Invalid Index. Please type an index between 1 and 14: 9

Choose an option:
L: List All Planes
M: List Planes by Make
A: Add a New Plane
R: Remove a Plane
Q: Save and Quit

L

Model              Make           Fuel Capacity  Empty Weight   Horsepower     Range          Cruise speed
------------------------------------------------------------------------------------------------------
1. 152                 Cessna         26             1081           110            414            106
2. 360                 Lancair        43             1090           180            990            208
3. C23 Sundowner       Beechcraft     57             1494           180            564            115
4. K35 Bonanza         Beechcraft     70             1832           250            534            168
```

```
5. M20R Ovation        Mooney         89             2205           280            969            189
6. Malibu Mirage       Piper          120            2435           350            1342           212
7. RV-12               Vans Aircraft  20             750            100            451            119
8. RV-9                Vans Aircraft  36             1057           160            616            163
9. Skyhawk 172         Cessna         53             1663           180            515            123
10. Super Cub           Piper          36             845            125            449            96
11. TB-21 GT Trinidad   Socata         88             1911           250            1025           168
12. Tiger               Grumman        51             1360           180            529            139
13. Tomahawk            Piper          30             1128           112            383            107

Choose an option:
L: List All Planes
M: List Planes by Make
A: Add a New Plane
R: Remove a Plane
Q: Save and Quit

Q

Data saved. Quitting the program.
```

4. Algorithmic Design

Before you begin coding, **you must first plan out the logic** and think about what data you will use to test your program for correctness. All programmers plan before coding - this saves a lot of time and frustration! Use the steps below to identify the inputs and outputs, calculations, and steps needed to solve the problem.

Use the pseudocode syntax shown in the document, supplemented with English phrases if necessary. **Do not include any implementation details (e.g. source code file names, class or struct definitions, or language syntax).** Do not include any C++ specific syntax or data types.

Algorithmic design:
a. Identify and list all of the user input variables and their data types. Include a variable name, data type, and description. Data types include string, integer, floating point, (single) character, and boolean. Data structures should be referenced by name, e.g. “array of integer” or “array of string”.
`fileName`

```
Data Type: array of char
Description: Stores the name of the airplane collection file entered
by the user.
choice

Data Type: char
Description: Stores the user's choice for the menu options (Add,
List, Remove, Search, List by Make, Quit).
input

Data Type: array of char
Description: Temporarily stores the user input for model and make of
the airplane when adding a new plane.
dVal

Data Type: floating point
Description: Temporarily stores the user input for fuel capacity of
the airplane when adding a new plane.
iVal

Data Type: integer
Description: Temporarily stores the user input for empty weight,
engine horsepower, range, and cruise speed of the airplane when
adding a new plane.
index

Data Type: integer
Description: Stores the index of the airplane to be removed, as
specified by the user.
make

Data Type: array of char
Description: Stores the make of the airplanes the user wants to list
when using the list by make functionality.

loadPlanes()

Prompts the user to enter the file name.
Uses: fileName
addAPlane()

Prompts the user to enter the model, make, fuel capacity, empty
weight, engine horsepower, range, and cruise speed of the airplane.
Uses: input, dVal, iVal
removeAPlane()
```

Prompts the user to enter the index of the airplane to remove.
Uses: index
listByMake()

Prompts the user to enter the make of the airplanes they want to list.
Uses: make
main Function
The main function interacts with the user to gather initial inputs and choices for the menu:

fileName
Used to get the name of the airplane collection file.
choice
Used to get the user's choice for the menu options

b. Identify and list all of the user output variables and their data types. Include a variable name, data type, and description. Data types include string, integer, floating point, (single) character, and boolean. Data structures should be referenced by name, e.g. “array of integer” or “array of string”.

fileName

Data Type: array of char
Description: Displays the name of the airplane collection file entered by the user when an error occurs (file not found).
choice

Data Type: char
Description: Displays the user's choice for the menu options (Add, List, Remove, Search, List by Make, Quit).
model

Data Type: array of char
Description: Displays the model of the airplane in the list and detailed views.
make

Data Type: array of char
Description: Displays the make of the airplane in the list and detailed views.
maxFuel

Data Type: floating point

```
Description: Displays the maximum fuel capacity of the airplane in the list
and detailed views.
emptyWeight

Data Type: integer
Description: Displays the empty weight of the airplane in the list and
detailed views.
engineHP

Data Type: integer
Description: Displays the engine horsepower of the airplane in the list and
detailed views.
maxRange

Data Type: integer
Description: Displays the maximum range of the airplane in the list and
detailed views.
cruiseSpeed

Data Type: integer
Description: Displays the cruise speed of the airplane in the list and
detailed views.
```

c. What calculations do you need to do to transform inputs into outputs? List all formulas needed, if applicable. If there are no calculations needed, state there are no calculations for this algorithm. Formulae should reference the variable names from step a and step b as applicable.

```
No calculations, only functions.
```

d. Design the logic of your program using pseudocode. Here is where you would use conditionals, loops or functions (if applicable) and list the steps in transforming inputs into outputs. Walk through your logic steps with the test data from the assignment document or the sample run above.

Use the syntax shown at the bottom of this document. Do not include any implementation details (e.g. file names) or C++ specific syntax.

```
---------- tools.h -----------------
DECLARE HEADER GUARD TOOLS_H

INCLUDE iostream
INCLUDE fstream
INCLUDE climits
INCLUDE cstring
INCLUDE iomanip

USE namespace std

DECLARE CONSTANTS
DECLARE integer STR_SIZE = 100
DECLARE integer ARR_SIZE = 100

DECLARE FUNCTION void welcome()

END HEADER GUARD TOOLS_H


---------- tools.cpp ---------------
INCLUDE "tools.h"

FUNCTION void welcome()
    DISPLAY "Welcome to the airplane collection program."
END FUNCTION

---------- plane.h ----------------
DECLARE HEADER GUARD PLANE_H

INCLUDE "tools.h"

USE namespace std

DECLARE CLASS Airplane
    PRIVATE:
        DECLARE char make[STR_SIZE]
        DECLARE char model[STR_SIZE]
        DECLARE double maxFuel
        DECLARE integer emptyWeight
        DECLARE integer engineHP
        DECLARE integer maxRange
        DECLARE integer cruiseSpeed
    PUBLIC:
        DECLARE CONSTRUCTOR Airplane()

        DECLARE FUNCTION const char* getMake()
        DECLARE FUNCTION const char* getModel()
        DECLARE FUNCTION double getMaxFuel()
        DECLARE FUNCTION integer getEmptyWeight()
        DECLARE FUNCTION integer getEngineHP()
        DECLARE FUNCTION integer getMaxRange()
        DECLARE FUNCTION integer getCruiseSpeed()

        DECLARE FUNCTION void setMake(const char*)
        DECLARE FUNCTION void setModel(const char*)
        DECLARE FUNCTION void setMaxFuel(double)
        DECLARE FUNCTION void setEmptyWeight(int)
        DECLARE FUNCTION void setEngineHP(int)
        DECLARE FUNCTION void setMaxRange(int)
        DECLARE FUNCTION void setCruiseSpeed(int)
END CLASS

END HEADER GUARD PLANE_H

–––––––––– plane.cpp ––––––––––––––
INCLUDE "plane.h"

CONSTRUCTOR Airplane()
    SET make = ""
    SET model = ""
    SET maxFuel = 0.0
    SET emptyWeight = 0
    SET engineHP = 0
    SET maxRange = 0
    SET cruiseSpeed = 0
END CONSTRUCTOR

FUNCTION const char* getMake()
    RETURN make
END FUNCTION

FUNCTION const char* getModel()
    RETURN model
END FUNCTION

FUNCTION double getMaxFuel()
    RETURN maxFuel
END FUNCTION

FUNCTION integer getEmptyWeight()
    RETURN emptyWeight
END FUNCTION

FUNCTION integer getEngineHP()
    RETURN engineHP
END FUNCTION

FUNCTION integer getMaxRange()
    RETURN maxRange
END FUNCTION

FUNCTION integer getCruiseSpeed()
    RETURN cruiseSpeed
END FUNCTION
```

```
FUNCTION void setMake(const char* newMake)
    COPY newMake TO make
END FUNCTION

FUNCTION void setModel(const char* newModel)
    COPY newModel TO model
END FUNCTION

FUNCTION void setMaxFuel(double newMaxFuel)
    SET maxFuel = newMaxFuel
END FUNCTION

FUNCTION void setEmptyWeight(integer newEmptyWeight)
    SET emptyWeight = newEmptyWeight
END FUNCTION

FUNCTION void setEngineHP(integer newEngineHP)
    SET engineHP = newEngineHP
END FUNCTION

FUNCTION void setMaxRange(integer newMaxRange)
    SET maxRange = newMaxRange
END FUNCTION

FUNCTION void setCruiseSpeed(integer newCruiseSpeed)
    SET cruiseSpeed = newCruiseSpeed
END FUNCTION

—--------- fleet.h —---------------
DECLARE HEADER GUARD FLEET_H

INCLUDE "plane.h"

DECLARE CLASS Fleet
    PRIVATE:
        DECLARE integer count
        DECLARE char fileName[STR_SIZE]
        DECLARE ifstream inFile
        DECLARE Airplane fleetAirplanes[ARR_SIZE]
        DECLARE Airplane tempPlane
        DECLARE FUNCTION bool insert()
    PUBLIC:
        DECLARE CONSTRUCTOR Fleet()
        DECLARE FUNCTION void setFileName(const char*)
        DECLARE FUNCTION integer loadPlanes()
        DECLARE FUNCTION void printPlanes()
        DECLARE FUNCTION void listByMake()
        DECLARE FUNCTION bool addAPlane()
        DECLARE FUNCTION bool removeAPlane()
        DECLARE FUNCTION void writePlanes()
END CLASS

END HEADER GUARD FLEET_H


--------- fleet.cpp ---------------
INCLUDE "fleet.h"
INCLUDE "tools.h"
INCLUDE cstring

CONSTRUCTOR Fleet()
    SET count = 0
    SET fileName = ""
END CONSTRUCTOR

FUNCTION void setFileName(const char* newFileName)
    COPY newFileName TO fileName
END FUNCTION

FUNCTION bool insert()
    IF count >= ARR_SIZE THEN
        RETURN false
    END IF
    DECLARE integer index
    FOR index = 0 TO count - 1 DO
        IF strcmp(fleetAirplanes[index].getModel(), tempPlane.getModel()) >
0 THEN
            BREAK
        END IF
    END FOR
    FOR integer i = count TO index + 1 STEP -1 DO
        SET fleetAirplanes[i] = fleetAirplanes[i - 1]
    END FOR
    SET fleetAirplanes[index] = tempPlane
    INCREMENT count
    RETURN true
END FUNCTION

FUNCTION integer loadPlanes()
    CALL inFile.open(fileName)
    IF NOT inFile.is_open() THEN
        DISPLAY fileName, " was not found. Try again or type 'quit' to exit
the program."
        RETURN -1
    END IF

    DECLARE char line[STR_SIZE]
    WHILE inFile.getline(line, STR_SIZE, ';') DO
        DECLARE Airplane plane
        CALL plane.setModel(line)

        CALL inFile.getline(line, STR_SIZE, ';')
        CALL plane.setMake(line)

        DECLARE double fuel
        CALL inFile >> fuel
        CALL inFile.ignore(1, ';')
        CALL plane.setMaxFuel(fuel)

        DECLARE integer weight, hp, range, speed
        CALL inFile >> weight
        CALL inFile.ignore(1, ';')
        CALL plane.setEmptyWeight(weight)

        CALL inFile >> hp
        CALL inFile.ignore(1, ';')
        CALL plane.setEngineHP(hp)

        CALL inFile >> range
        CALL inFile.ignore(1, ';')
        CALL plane.setMaxRange(range)

        CALL inFile >> speed
        CALL inFile.ignore(1, '\n')
        CALL plane.setCruiseSpeed(speed)

        SET tempPlane = plane
        CALL insert()
    END WHILE

    CALL inFile.close()
    RETURN count
END FUNCTION

FUNCTION void printPlanes()
    DISPLAY "Model", "Make", "Fuel Capacity", "Empty Weight", "Horsepower",
"Range", "Cruise speed"
    DISPLAY
"---------------------------------------------------------------------------
-------------------------"
    FOR integer i = 0 TO count - 1 DO
        DISPLAY i + 1, ". ", fleetAirplanes[i].getModel(),
fleetAirplanes[i].getMake(), fleetAirplanes[i].getMaxFuel(),
fleetAirplanes[i].getEmptyWeight(), fleetAirplanes[i].getEngineHP(),
fleetAirplanes[i].getMaxRange(), fleetAirplanes[i].getCruiseSpeed()
    END FOR
END FUNCTION

FUNCTION void listByMake()
    DECLARE char make[STR_SIZE]
    DISPLAY "Please type the make of the airplanes you would like to list: 
"
    CALL cin.getline(make, STR_SIZE)

    DISPLAY "Model", "Make", "Fuel Capacity", "Empty Weight", "Horsepower", 
"Range", "Cruise speed"
    DISPLAY 
"---------------------------------------------------------------------------
---------------------------"
    FOR integer i = 0 TO count - 1 DO
        IF strcmp(fleetAirplanes[i].getMake(), make) == 0 THEN
            DISPLAY i + 1, ". ", fleetAirplanes[i].getModel(), 
fleetAirplanes[i].getMake(), fleetAirplanes[i].getMaxFuel(), 
fleetAirplanes[i].getEmptyWeight(), fleetAirplanes[i].getEngineHP(), 
fleetAirplanes[i].getMaxRange(), fleetAirplanes[i].getCruiseSpeed()
        END IF
    END FOR
END FUNCTION

FUNCTION bool addAPlane()
    DECLARE Airplane plane
    DECLARE char input[STR_SIZE]
    DECLARE double dVal
    DECLARE integer iVal

    DISPLAY "What is the model (name) of the airplane? "
    CALL cin.getline(input, STR_SIZE)
    CALL plane.setModel(input)

    DISPLAY "What is the make (manufacturer) of the airplane? "
    CALL cin.getline(input, STR_SIZE)
    CALL plane.setMake(input)

    DISPLAY "What is the fuel capacity in gallons? "
    CALL cin >> dVal
    WHILE cin.fail() OR dVal <= 0.0 OR dVal > 150.0 DO
        CALL cin.clear()
        CALL cin.ignore(INT_MAX, '\n')
        DISPLAY "Please enter a decimal number for fuel capacity between 
1.00 and 150.00: "
        CALL cin >> dVal
    END WHILE
    CALL plane.setMaxFuel(dVal)
    CALL cin.ignore(INT_MAX, '\n')
    DISPLAY "What is the empty weight? "
    CALL cin >> iVal
    WHILE cin.fail() OR iVal <= 0 OR iVal > 3000 DO
        CALL cin.clear()
        CALL cin.ignore(INT_MAX, '\n')
        DISPLAY "The weight must be a whole number between 1 and 3000
pounds: "
        CALL cin >> iVal
    END WHILE
    CALL plane.setEmptyWeight(iVal)
    CALL cin.ignore(INT_MAX, '\n')

    DISPLAY "What is the horsepower of the engine? "
    CALL cin >> iVal
    WHILE cin.fail() OR iVal <= 0 OR iVal > 400 DO
        CALL cin.clear()
        CALL cin.ignore(INT_MAX, '\n')
        DISPLAY "The horsepower must be a whole number between 1 and 400: "
        CALL cin >> iVal
    END WHILE
    CALL plane.setEngineHP(iVal)
    CALL cin.ignore(INT_MAX, '\n')

    DISPLAY "What is the range? "
    CALL cin >> iVal
    WHILE cin.fail() OR iVal <= 0 OR iVal > 2000 DO
        CALL cin.clear()
        CALL cin.ignore(INT_MAX, '\n')
        DISPLAY "The range must be a whole number between 1 and 2000: "
        CALL cin >> iVal
    END WHILE
    CALL plane.setMaxRange(iVal)
    CALL cin.ignore(INT_MAX, '\n')

    DISPLAY "What is the cruise speed? "
    CALL cin >> iVal
    WHILE cin.fail() OR iVal <= 0 DO
        CALL cin.clear()
        CALL cin.ignore(INT_MAX, '\n')
        DISPLAY "The cruise speed must be a positive whole number: "
        CALL cin >> iVal
    END WHILE
    CALL plane.setCruiseSpeed(iVal)
    CALL cin.ignore(INT_MAX, '\n')

    SET tempPlane = plane
    RETURN insert()
END FUNCTION
```

```
FUNCTION bool removeAPlane()
    DECLARE integer index
    DISPLAY "Which index would you like to remove (1 - ", count, ")? "
    CALL cin >> index
    WHILE cin.fail() OR index < 1 OR index > count DO
        CALL cin.clear()
        CALL cin.ignore(INT_MAX, '\n')
        DISPLAY "Invalid Index. Please type an index between 1 and ",
count, ": "
        CALL cin >> index
    END WHILE
    DECREMENT index

    FOR integer i = index TO count - 2 DO
        SET fleetAirplanes[i] = fleetAirplanes[i + 1]
    END FOR
    DECREMENT count
    RETURN true
END FUNCTION

FUNCTION void writePlanes()
    DECLARE ofstream outFile(fileName)
    FOR integer i = 0 TO count - 1 DO
        CALL outFile << fleetAirplanes[i].getModel() << ";"
        CALL outFile << fleetAirplanes[i].getMake() << ";"
        CALL outFile << fleetAirplanes[i].getMaxFuel() << ";"
        CALL outFile << fleetAirplanes[i].getEmptyWeight() << ";"
        CALL outFile << fleetAirplanes[i].getEngineHP() << ";"
        CALL outFile << fleetAirplanes[i].getMaxRange() << ";"
        CALL outFile << fleetAirplanes[i].getCruiseSpeed() << endl
    END FOR
    CALL outFile.close()
END FUNCTION

--------- main.cpp ---------------
INCLUDE "tools.h"
INCLUDE "fleet.h"
INCLUDE cstring

FUNCTION main()
    CALL welcome()
    DECLARE Fleet myFleet
    DECLARE char fileName[STR_SIZE]
    DECLARE bool fileLoaded = false

    WHILE NOT fileLoaded DO
        DISPLAY "What is the name of the airplane collection file? "
        CALL cin.getline(fileName, STR_SIZE)
        IF strcmp(fileName, "quit") == 0 THEN
            RETURN 0
        END IF

        CALL myFleet.setFileName(fileName)
        IF myFleet.loadPlanes() != -1 THEN
            SET fileLoaded = true
        END IF
    END WHILE

    DECLARE char choice
    DO
        DISPLAY "What would you like to do? (A)dd a plane, (L)ist all
planes, (R)emove a plane by index, (S)earch for a plane, List all planes by
(M)ake, or (Q)uit? "
        CALL cin >> choice
        CALL cin.ignore(INT_MAX, '\n')

        SWITCH choice
            CASE 'A'
            CASE 'a'
                IF myFleet.addAPlane() THEN
                    DISPLAY "Successfully added the plane to the database."
                ELSE
                    DISPLAY "Failed to add the plane to the database."
                END IF
                BREAK
            CASE 'L'
            CASE 'l'
                CALL myFleet.printPlanes()
                BREAK
            CASE 'R'
            CASE 'r'
                IF myFleet.removeAPlane() THEN
                    DISPLAY "Successfully removed the plane from the
database."
                ELSE
                    DISPLAY "Failed to remove the plane from the database."
                END IF
                BREAK
            CASE 'S'
            CASE 's'
                // Search functionality can be implemented here
                BREAK
            CASE 'M'
            CASE 'm'
                CALL myFleet.listByMake()
                BREAK
            CASE 'Q'
            CASE 'q'
                CALL myFleet.writePlanes()
                DISPLAY "Database file updated. Terminating Program."
                BREAK
            DEFAULT:
                DISPLAY "Invalid option. Please try again."
        END SWITCH
    WHILE choice != 'Q' AND choice != 'q'

    RETURN 0
END FUNCTION
```

5. Pseudocode Syntax

Think about each step in your algorithm as an action and use the verbs below:

To do this:	**Use this verb:**	**Example:**
Create a variable	DECLARE	`DECLARE integer num_dogs`
Print to the console window	DISPLAY	`DISPLAY “Hello!”`
Read input from the user into a variable	INPUT	`INPUT num_dogs`
Update the contents of a variable	SET	`SET num_dogs = num_dogs + 1`
Conditionals		

Use a single alternative conditional	IF *condition* THEN
	    statement
	    statement
	END IF
		`IF num_dogs > 10 THEN`
		    `DISPLAY “That is a lot of dogs!”`
		`END IF`

Use a dual alternative conditional	IF *condition* THEN
	    statement
	    statement
	ELSE
	    statement
	    statement
	END IF
		`IF num_dogs > 10 THEN`
		    `DISPLAY “You have more than 10 dogs!”`
		`ELSE`
		    `DISPLAY “You have ten or fewer dogs!”`
		`END IF`

Use a switch/case statement	SELECT *variable or expression*
	  CASE *value_1:*
	    statement
	    statement
	  CASE *value_2:*
	    statement
		`SELECT num_dogs`
		   `CASE 0: DISPLAY “No dogs!”`
		   `CASE 1: DISPLAY “One dog..”`
		   `CASE 2: DISPLAY “Two dogs..”`
		   `CASE 3: DISPLAY “Three dogs..”`
		   `DEFAULT: DISPLAY “Lots of dogs!”`

	statement CASE *value_2:* *statement* *statement* DEFAULT: *statement* *statement* END SELECT	`END SELECT`
Loops		

Loop while a condition is true - the loop body will execute 0 or more times.	WHILE *condition*
	statement
	statement
	END WHILE
		`SET num_dogs = 1`
		`WHILE num_dogs < 10`
		`   DISPLAY num_dogs, " dogs!"`
		`   SET num_dogs = num_dogs + 1`
		`END WHILE`

Loop while a condition is true - the loop body will execute 1 or more times.	DO
	statement
	statement
	WHILE *condition*
		`SET num_dogs = 1`
		`DO`
		`   DISPLAY num_dogs, " dogs!"`
		`   SET num_dogs = num_dogs + 1`
		`WHILE num_dogs < 10`

Loop a specific number of times.	FOR *counter* = *start* TO *end*
	statement
	statement
	END FOR
		`FOR count = 1 TO 10`
		`   DISPLAY num_dogs, " dogs!"`
		`END FOR`

Functions		

Create a function	FUNCTION *return_type name (parameters)*
	statement
	statement
	END FUNCTION
		`FUNCTION Integer add(Integer num1, Integer num2)`
		`   DECLARE Integer sum`
		`   SET sum = num1 + num2`
		`   RETURN sum`
		`END FUNCTION`

Call a function	CALL *function_name*	`CALL add(2, 3)`
Return data from a function	RETURN *value*	`RETURN 2 + 3`

Airplane (Class 1): similar to struct from A01 w/ encapsulated member functions

Collection of Airplanes (Class 2): Fleet or AirplaneCollection [must be a capital letter]

```
Airplane {
    [all member variables(properties) must be private here]
 private:
        make
        model
        maximum fuel
        empty weight
        horsepower
        cruise speed

}
```

```
Fleet {
    [include Airplane nested]
    [all member variables(properties) must be private here]
private:
    -   int count;
    -   char fileName[STR_SIZE]
    -   ifstream inFile;
    -   Airplane fleetAirplanes[ARR_SIZE]
    -   bool insert(); // private method
public:
    -   Fleet(); // Default constructor
    -   int loadPlanes();
    -   void printPlanes();
    -   void listByMake();
    -   bool addAPlane();
    -   bool removeAPlane();
    -   void writePlanes();
}
```