PROGRAM 4 / CSC 1300

IMPERIAL MECHANIC

WHAT TO TURN IN:

- Zip & upload to PROGRAM 4 folder in ilearn dropbox
- **Prog4.cpp**, **prog4.h**, **functions.cpp**
- **vehicles.txt** – this should be a text file with at least FIVE Vehicles that you created with your program (**data should be separated by #**).

DESCRIPTION:

Your assignment is to write a program for a vehicle mechanic. The mechanic has a garage(array) that can hold up to 100 vehicles. Information about each vehicle needs will be kept and manipulated in this program. The user should be able to load vehicle information from any file he or she chooses, add vehicles manually, delete a vehicle, print vehicle information to either a file or to the screen, or print a cost analysis of each vehicle and the total cost in parts, work hours and supplies repair these vehicles.

WHAT'S NEW IN PROGRAM 5:

- Structures (structure array, nested structures)

PROGRAM SPECIFICATIONS

STRUCTURES: (DEFINE ALL STRUCTURES IN PROG5.H)

You will need to create two structures.

Cost will have the following members and they should all be defined as **float**:

- The number of hours it takes to repair a specific **Vehicle**.
- The cost (per hour) for repairing this **Vehicle**.
- The cost of parts for this **Vehicle**.
- The cost of materials/supplies for this **Vehicle**.

Vehicles will have the following members:

- The name of the vehicle
- The description of the vehicle
- If the vehicle has weapons (Boolean variable – **true** or **false**)
- A variable to hold the **Cost** structure members (nested structure)

THE MAIN FUNCTION: (DEFINE MAIN FUNCTION IN PROG5.CPP)

You will need to create an array of 100 elements of the **Vehicles** data type. You will also need to create a variable that will keep up with the current number of **Vehicles**.

The main function will display a menu of five options:

```
        1.  Enter some vehicles.
        2.  Delete a vehicle.
        3.  Print vehicles.
        4.  Print statistics on vehicle cost.
        5.  End program.
```

Make sure you always validate all user choices in the whole program before proceeding to do what the user wants. Each menu choice will call a function that you will create.

If the user chooses option 1, then your program will call the **enterVehicle** function.

If the user chooses option 2, then your program will call the **deleteVehicle** function.

If the user chooses option 3, then your program will call the **printVehicles** function.

If the user chooses option 4, then your program will call the **printStatistics** function.

If the user chooses option 5, then your program will ask the user if they wish to save their **Vehicles** list to a file. If they choose yes, then your program should call the **saveVehiclesToFile** function and then end. If they choose no, then your program should just end.

ENTERVEHICLES FUNCTION (DEFINE ALL OTHER FUNCTIONS IN FUNCTIONS.CPP)

The **enterVehicles**() function takes two parameters: the number of **Vehicles** currently loaded in the **Vehicles** array and the **Vehicles** array. The function will return the new number of **Vehicles**. Before trying to add **any Vehicles**, this function should first check to make sure the number of **Vehicles** isn't already 100. Because if it is, then your program should not add any Vehicles, but should instead tell the user that their garage is at full capacity and that they are not able to add **Vehicles**. Then the function should end.

Otherwise, your program will display a menu asking the user if they would like to do one of the following:

```
1. Load my Vehicles from a file.
2. Enter one Vehicle manually.
```

Validate the user's choice.

If the user chooses option 1, then ask the user what the name of the file is that they would like to load the **Vehicles** from. Then open their file. Check if the file could open before reading from it.

Read each vehicle's data from the file and place them in the **Vehicles** array, making sure that you increment the number of vehicles each time a vehicle is added. When you are reading from the file, everything read in will have to be read in as a **string**. Some of the **Vehicles** members are strings, so that won't be a problem. However, some of the Vehicle members are **float**. So, this function will have to call the **convertToFloat**() function (provided later) in order to convert the **string** to a **float** and then placed in the **Vehicles** array. Then return the number of **Vehicles** at the end of the function.

```
What would you like to do?
        1.  Enter some vehicles.
        2.  Delete a vehicle.
        3.  Print vehicles.
        4.  Print statistics on vehicle cost.
        5.  End program.
        Enter 1, 2, 3, 4, or 5.
CHOICE:  1

What do you want to do?
        1.  Load my vehicles from a file.
        2.  Enter one vehicle manually.
        Choose 1 or 2.
CHOICE:  1

What is the name of the file with your list of vehicles? (ex: filename.txt)
FILENAME:  vehicle.txt


All vehicles from vehicle.txt have been added to the program.
```

If the user chooses option 2, then you will want to start a loop so that the user can add multiple vehicles manually without returning to the main menu.

Ask the user for the following:
(make sure you place each bit of information in the correct place in the `Vehicles` array)

- NAME:
- DESCRIPTION:
- DOES THE VEHICLE HAVE WEAPONS? (y or n)
- How many hours do you spend repairing the [insert Vehicle name here]?
  NUM HOURS:
- What is the cost per hour for repairing the [insert Vehicle name here]?
  COST PER HOUR: $
- How much money do you spend on parts for the [insert Vehicle name here]?
  PARTS COST: $
- How much money do you spend on other supplies for the [insert Vehicle name here]?
  SUPPLY COST: $

Then, increment the number of `Vehicles` by one. Then, ask the user if they want to add another vehicle. If they do, then repeat this option. If not, then just return the number of `Vehicles`.

```
What would you like to do?
        1.  Enter some vehicles.
        2.  Delete a vehicle.
        3.  Print vehicles.
        4.  Print statistics on vehicle cost.
        5.  End program.
        Enter 1, 2, 3, 4, or 5.
CHOICE:  1

What do you want to do?
        1.  Load my vehicles from a file.
        2.  Enter one vehicle manually.
        Choose 1 or 2.
CHOICE:  2


NAME: Speeder Bike

DESCRIPTION:  Speeder bikes, also known as jumpspeeders, were open-air
repulsorlift vehicles that emphasized speed and maneuverability over stability.
The Jedi used the Undicur-class jumpspeeder, which proved popular with civilians.

DOES THIS VEHICLE HAVE WEAPONS? (y or n):  y


How many hours do you spend repairing the Speeder Bike?
NUM HOURS:  9

What is the cost per hour for repairing for the Speeder Bike?
COST PER HOUR:  $2.00

How much money do you spend on part for the Speeder Bike?
PART COST:  $3.45
```

```
How much money do you spend on supplies for the Speeder Bike?
SUPPLY COST:  $9384.56

The Speeder Bike has been added.

Want to add more vehicles? (y or n)  n
```

DELETE VEHICLE FUNCTION

The **`deleteVehicle`**() function has two parameters: the current number (`int`) of **`Vehicles`** in the **`Vehicles`** array and the **`Vehicles`** array. This function returns the new number (`int`) of vehicles in the **`Vehicles`** array.

First, this function will print

```
The following is a list of all the vehicles you take care of:
```

Then it will print the name of each vehicle. Then, your program will ask the user:

```
What vehicle do you wish to remove?
VEHICLE NAME:
```

Your program should then read in the name and place it in a variable.

Then, this function will call the **`moveArrayElements`**() function, which will take care of removing the **`Vehicle`**. The **`moveArrayElements`** function returns a `bool` (`true` or `false`) value to tell if the vehicle was removed or not.

Check to see if the vehicle was removed. If it was, then decrement the number of **`Vehicles`** and print out "You have removed [insert vehicle name here]."

Then return the new number of **`Vehicles`**.

```
What would you like to do?
        1.  Enter some vehicles.
        2.  Delete a vehicle.
        3.  Print vehicles.
        4.  Print statistics on vehicle cost.
        5.  End program.
        Enter 1, 2, 3, 4, or 5.
CHOICE:  2

The following is a list of all the vehicles you take care of:
AT-AT
Podracer
TIE Fighter
TIE Interceptor
Star Destroyer
Speeder Bike

What vehicle do you wish to remove?
VEHICLE NAME: TIE Interceptor

You have removed TIE Interceptor.


What would you like to do?
```

```
        1.  Enter some vehicles.
        2.  Delete a vehicle.
        3.  Print vehicles.
        4.  Print statistics on vehicle cost.
        5.  End program.
        Enter 1, 2, 3, 4, or 5.
CHOICE:  2

The following is a list of all the vehicles you take care of:
AT-AT
Podracer
TIE Fighter
Star Destroyer
Speeder Bike


What vehicle do you wish to remove?
VEHICLE NAME: Mario Kart

Sorry, a vehicle by the name Mario Kart could not be found.
```

MOVEARRAYELEMENTS FUNCTION

The **`moveArrayElements()`** function has the following parameters: a **`string`** with the name of the vehicle that the user wishes to remove, the current number (`int`) of **`Vehicles`** in the **`Vehicles`** array, and the **`Vehicles`** array. This function returns a `bool` (`true` or `false`) variable that is `true` if the vehicle was removed and `false` if it was not removed.

This function should first find the index number of the vehicle that needs to be removed. Once that is found, then you know there is a vehicle in the **`Vehicles`** array by that name and that your program will be able to remove it. If your program cannot find the vehicle in the array, then you return `false` from this function. Otherwise, this function should now overwrite the element with the Vehicle to delete (**`i`**) with the next element in the array (**`i +1`**), moving each element after the deleted element to the left. Then return `true` that the vehicle was found & removed.

PRINT VEHICLES FUNCTION

The **`printVehicles()`** function is a **`void`** function and contains the following parameters: number (`int`) of vehicles currently in the **`Vehicles`** array and the **`Vehicles`** array. The **`printVehicles()`** function will first display a menu to the user containing the following options:

```
1. Print vehicles to the screen.
2. Print vehicles to a file.
```

Validate the user's choice.

If the user chooses option 1, then you will print out all the vehicles in the **`Vehicles`** array to the screen in the following format:

```
What would you like to do?
        1. Print vehicle to the screen.
        2. Print vehicle to a file.
        Choose 1 or 2.
CHOICE:  1
------------------------------------------------------------------------
```

```
VEHICLE 1:
Name:           AT-AT
Description:
         The All Terrain Armored Transport, or AT-AT walker, is a four-legged
         transport and combat vehicle used by the Imperial ground
         forces.

Has weapons?    yes
Number of Hours to repair the Vehicle:  100.00
Cost Per Hour:  $ 5000.00
Cost for Parts: $ 1000.00
Supplies Cost:  $ 1000.00


------------------------------------------------------------------------
VEHICLE 2:
Name:           Podracer
Description:
         Capable of achieving speeds over 700 kilometers per hour, podracers
         were used in the similarly named sport of podracing.

Has weapons?    no
Number of Hours to repair the Vehicle:  24.00
Cost Per Hour:  $ 500.00
Cost for Parts: $ 1000.00
Supplies Cost:  $ 200.00


------------------------------------------------------------------------
VEHICLE 3:
Name:           TIE Fighter
Description:
         Propelled by Twin Ion Engines, TIE fighters are fast, agile,
         yet fragile starfighters produced by Sienar Fleet Systems for
         the Galactic Empire.

Has weapons?    yes
Number of Hours to repair the Vehicle:  50.00
Cost Per Hour:  $ 75000.00
Cost for Parts: $ 75000.00
Supplies Cost:  $ 75000.00


------------------------------------------------------------------------
VEHICLE 4:
Name:           TIE Intercepter
Description:
         Faster, more maneuverable, and far more lethal than the standard
         TIE Fighter.

Has weapons?    yes
Number of Hours to repair the Vehicle:  50.00
Cost Per Hour:  $ 75000.00
Cost for Parts: $ 75000.00
Supplies Cost:  $ 75000.00
```

```
------------------------------------------------------------------------
VEHICLE 5:
Name:           Star Destroyer
Description:
         A single Star Destroyer could project considerable influence
         over a solar system in the name of the Empire: each can be deployed
         individually as both a forward operating base and as
         mobile weapon systems platform responsible for safeguarding
         multiple planets, trade routes and systems, and carried enough
         firepower to subdue an entire planetary system or annihilate
         a small rebel fleet.

Has weapons?    yes
Number of Hours to repair the Vehicle:  1000.00
Cost Per Hour:  $ 1000.00
Cost for Parts: $ 2000.00
Supplies Cost:  $ 30000.00
```

Notice that the description does word wrapping. To clarify, each word is not split up – they are kept together.

You get a bonus of 5 points if you can do word wrapping for your descriptions.

If the user selects option 2, then your program should ask the user what is the name of the file that they wish to print the vehicles to. Then, open this file. You do not have to check for errors for this file because it probably will not yet exist. Then, write all the information to the file like you did to the screen if the user had chosen option **1**.

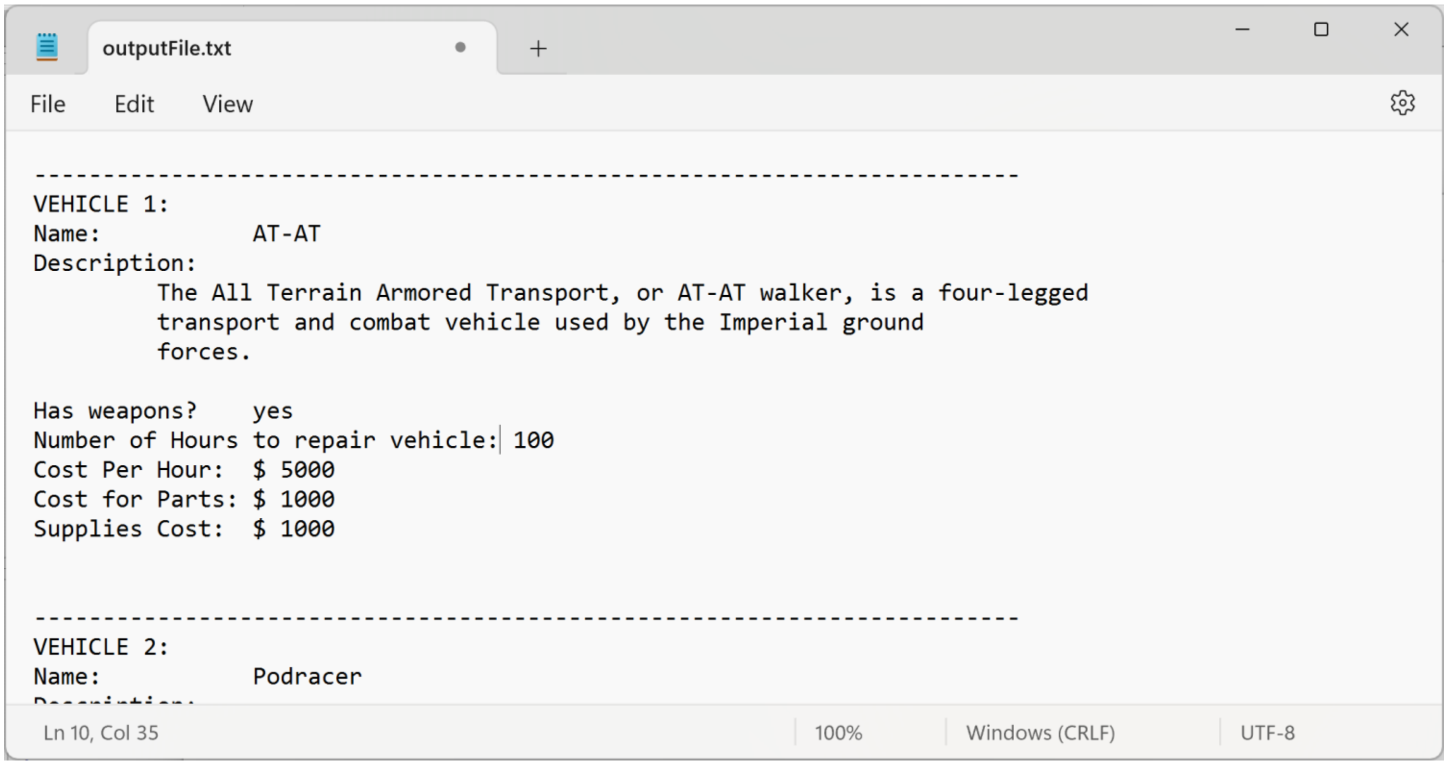
Then, your program should write “Your Vehicles have been written to [insert filename here].”

```
What would you like to do?
        1.  Enter some vehicles.
        2.  Delete a vehicle.
        3.  Print vehicles.
        4.  Print statistics on vehicle cost.
        5.  End program.
        Enter 1, 2, 3, 4, or 5.
CHOICE:  3

What would you like to do?
        1. Print Vehicle to the Screen.
        2. Print Vehicle to a file.
        Choose 1 or 2.
CHOICE:  2
What is the name of your file you wish to write to?
FILENAME:  outputFile.txt

Your vehicles have been entered into outputFile.txt.
```

Here is a screen capture of the beginning of the outputFile.txt file:



```
-----------------------------------------------------------------------
VEHICLE 1:
Name:           AT-AT
Description:
         The All Terrain Armored Transport, or AT-AT walker, is a four-legged
         transport and combat vehicle used by the Imperial ground
         forces.

Has weapons?    yes
Number of Hours to repair vehicle: 100
Cost Per Hour:  $ 5000
Cost for Parts: $ 1000
Supplies Cost:  $ 1000


-----------------------------------------------------------------------
VEHICLE 2:
Name:           Podracer
```

PRINTSTATISTICS FUNCTION

The **`printStatistics()`** function is a void function and it contains two parameters: the current number (**`int`**) of vehicles in the **`Vehicles`** array and the **`Vehicles`** array.

This function should print out the total cost of each vehicle in a table. To figure out the cost of a vehicle, use the formula:

`cost = numHours * costPerHour + partCost + materialCost`

Then, this function should also keep a running total of the total cost of all the vehicles.

```
What would you like to do?
        1.  Enter some vehicles.
        2.  Delete a vehicle.
        3.  Print vehicles.
        4.  Print statistics on vehicle cost.
        5.  End program.
        Enter 1, 2, 3, 4, or 5.
CHOICE:   4

COST OF EACH VEHICLE:

VEHICLE                          COST
AT-AT              $      502000.00
Podracer           $       13200.00
TIE Fighter        $     3900000.00
TIE Intercepter    $     3900000.00
Star Destroyer     $     1032000.00

TOTAL COST:        $     9347200.00
```

SAVEVEHICLESTOFILE FUNCTION

The **saveVehiclesToFile()** function is a void function and it contains two parameters: the current number (**int**) of Vehicles in the **Vehicles** array and the **Vehicles** array. The function should ask the user what the name of the file that they wish to save their Vehicles to. The function should then open that file and write out all the vehicle data in the following order:

- Name
- Description
- Weapons
- Number of hours to care for vehicle
- Cost per hour
- Cost of parts
- Supplies cost

There should be no newlines or endlines in the file. **All data should be separated by a pound sign (#)** instead of a space.

After printing all data from the Vehicles array to the file, this function should print out a message to standard output saying "Your Vehicles were successfully saved to the [insert filename here] file."

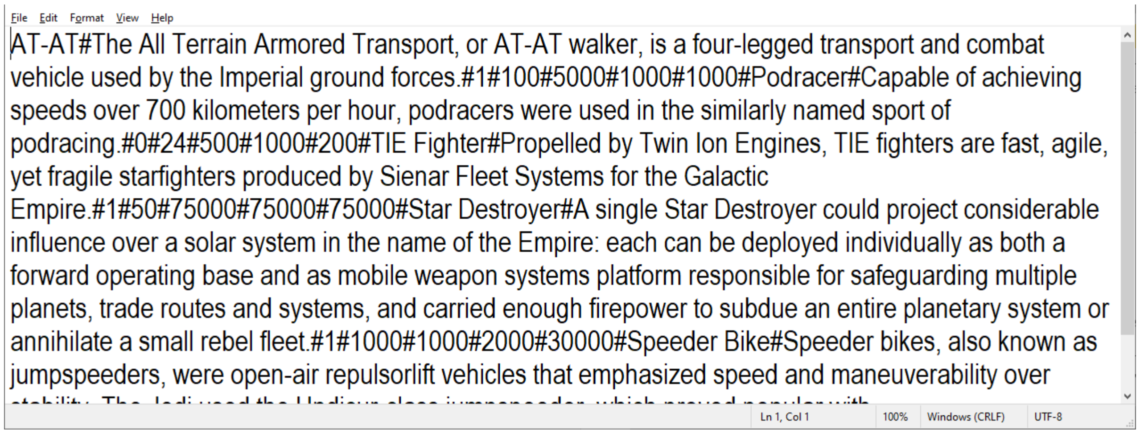
```
What would you like to do?
        1.  Enter some vehicles.
        2.  Delete a vehicle.
        3.  Print vehicles.
        4.  Print statistics on vehicle cost.
        5.  End program.
        Enter 1, 2, 3, 4, or 5.
CHOICE:  5


Would you like to save your vehicle list to a file? (y or n)  y
What is the name of the file you want to save your vehicles to?
FILENAME:  11-02-2023prog4dump.txt

Your vehicles were successfully saved to the 11-02-2023prog4dump.txt file.


GOODBYE!
```

Here is a screen capture of the beginning of the **11-02-2023prog4dump.txt** text file:



```
AT-AT#The All Terrain Armored Transport, or AT-AT walker, is a four-legged transport and combat
vehicle used by the Imperial ground forces.#1#100#5000#1000#1000#Podracer#Capable of achieving
speeds over 700 kilometers per hour, podracers were used in the similarly named sport of
podracing.#0#24#500#1000#200#TIE Fighter#Propelled by Twin Ion Engines, TIE fighters are fast, agile,
yet fragile starfighters produced by Sienar Fleet Systems for the Galactic
Empire.#1#50#75000#75000#75000#Star Destroyer#A single Star Destroyer could project considerable
influence over a solar system in the name of the Empire: each can be deployed individually as both a
forward operating base and as mobile weapon systems platform responsible for safeguarding multiple
planets, trade routes and systems, and carried enough firepower to subdue an entire planetary system or
annihilate a small rebel fleet.#1#1000#1000#2000#30000#Speeder Bike#Speeder bikes, also known as
jumpspeeders, were open-air repulsorlift vehicles that emphasized speed and maneuverability over
stability. The Jedi used the Undiur-class jumpspeeder, which proved popular with
```

CONVERTTOFLOAT FUNCTION

The **`convertToFloat()`** function will need to be used when reading data from a file and inputting the data into the **`Vehicles`** array (so this function will be called from the **`enterVehicles`** function). I am providing this function for you. You just need to insert it into your program so you can use it.

First, make sure to include the following header file to be able to use this function:

```
#include <sstream>
```

Then, put the following function at the bottom of your **`functions.cpp`** source file:

```
float convertToFloat(string s)
{
	istringstream i(s);
	float x;
	if (!(i >> x))
		x = 0;
	return x;
}
```