PROGRAM 5 / CSC 1300

# Monster Island

 **https://www.uhdpaper.com/2019/04/godzilla-vs-king-ghidorah-godzilla-king\_22.html**

## Assignment Date: 4/15/2020

## Due Date: 4/29/2020 (by 11:59pm) NO LATE SUBMISSIONS!!

### What to Turn In:

* Zip & upload to PROGRAM 5 folder in ilearn dropbox
* **prog5.cpp**, **prog5.h**, **functions.cpp**
* **Monsters.txt** – this should be a text file with at least FIVE Monsters that you created with your program (data should be separated by #).

### Description:

Monster island is a remote island in which humans can house giant monsters and care for them so that they don’t go on a rampage. In this program you will create a way for people to keep track of the information about each monster on the island. The island can hold up to 100 Monsters. Information about each Monster needs will be kept and manipulated in this program. The user should be able to load Monster information from any file he or she chooses, add Monsters manually, delete a Monster, print Monster information to either a file or to the screen, or print a cost analysis of each Monster and the total cost to house and take care of these Monsters.

### 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. One is called **Cost**. Cost will have the following members:

* The number of hours it takes to take care of a specific Monster.
* The cost (per hour) of taking care of this Monster.
* The cost of food to feed this Monster for one week.
* The cost of materials/supplies (grooming, medical) for this Monster for one week

The second structure is called **Monsters**. Monsters will have the following members:

* The name of the Monster
* The description of the Monster
* The average length of the Monster (in feet)
* The average height of the Monster (in feet)
* The location of the Monster (example: its origin)
* If the Monster is dangerous (Boolean variable)
* A variable to hold the Cost structure members

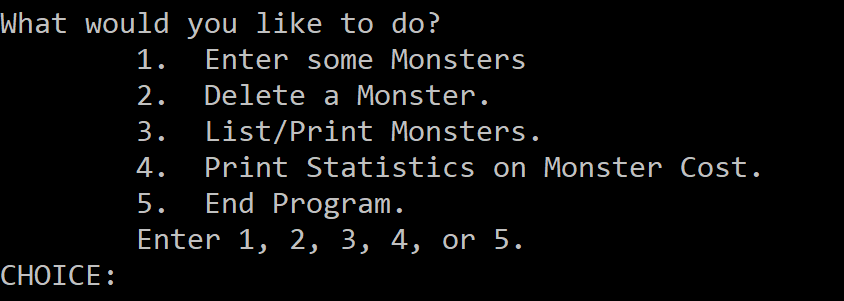
### The Main Function: (define main function in Prog5.cpp)

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

The main function will display a menu of five options:

1. Enter some Monsters.
2. Delete a Monster
3. List/Print Monsters.
4. Print Monster Costs.
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 enterMonster function. If the user chooses option 2, then your program will call the deleteMonster function. If the user chooses option 3, then your program will call the printMonsters 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 Monster list to a file. If they choose yes, then your program should call the saveMonstersToFile function and then end. If they choose no, then your program should just end.

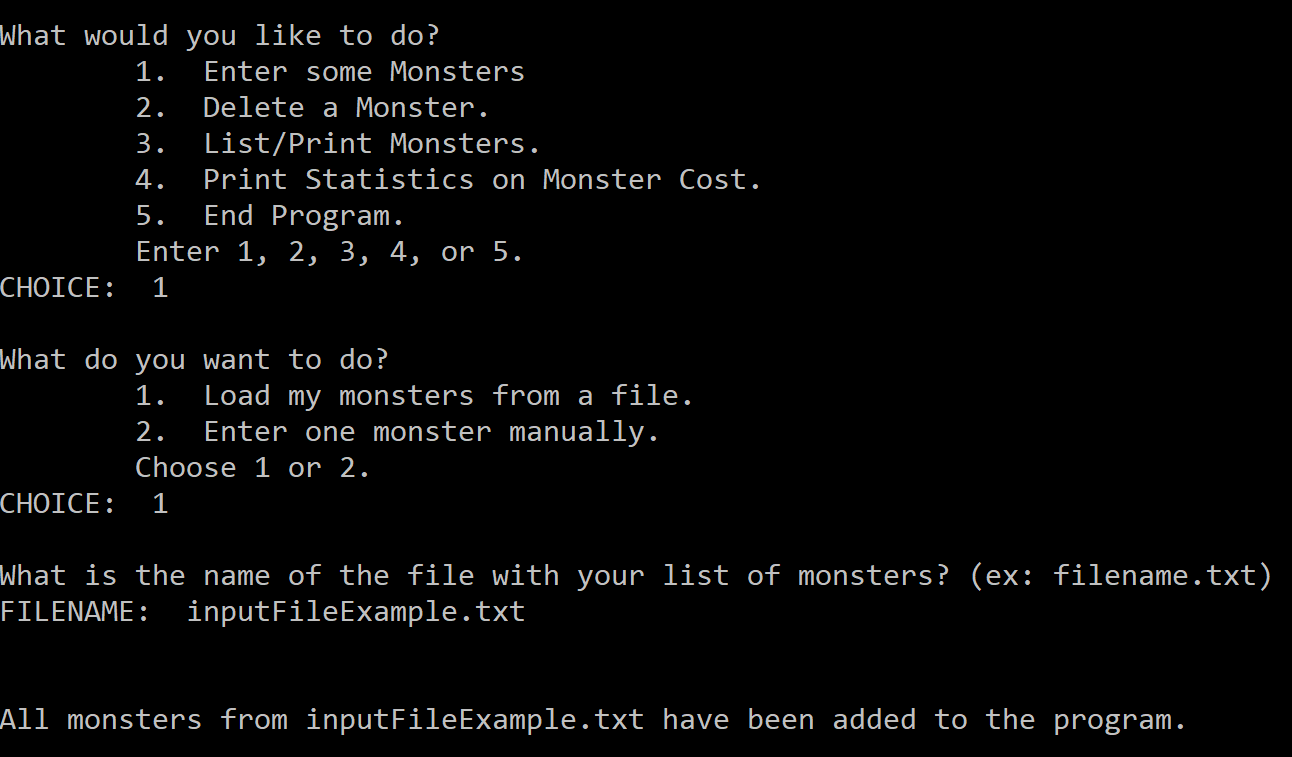
### enterMonsters function (define all other functions in functions.cpp)

The enterMonsters() function takes two parameters: the number of Monsters currently loaded in the Monsters array and the Monsters array. The function will return the new number of Monsters. Before trying to add any Monsters, this function should first check to make sure the number of Monsters isn’t already 100. Because if it is, then your program should not add any Monsters, but should instead tell the user that their island is at full capacity and that they are not able to add Monsters. 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 Monsters from a file.
2. Enter one Monster 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 Monsters from. Then open their file. Check if the file could open before reading from it.

Read each Monster from the file and place them in the Monsters array, making sure that you increment the number of Monsters each time a Monster is added. When you are reading from the file, everything read in will have to be read in as a string. Some of the Monster members are strings, so that won’t be a problem. However, some of the Monster members are floats. 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 Monsters array. Then return the number of Monsters at the end of the function.

If the user chooses option 2, then you will want to start a loop so that the user can add multiple Monsters 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 Monsters array)

* NAME:
* DESCRIPTION:
* AVERAGE LENGTH (in feet):
* AVERAGE HEIGHT (in feet):
* LOCATION:
* IS IT A DANGEROUS MONSTER? (y or n)
* How many hours do you spend caring for the [insert Monster name here]?  
  NUM HOURS:
* What is the cost per hour for caring for the [insert Monster name here]?  
  COST PER HOUR:
* How much money do you spend on food for the [insert Monster name here]?  
  FOOD COST:
* How much money do you spend on grooming and medical supplies for the [insert Monster name here]?  
  SUPPLY COST:

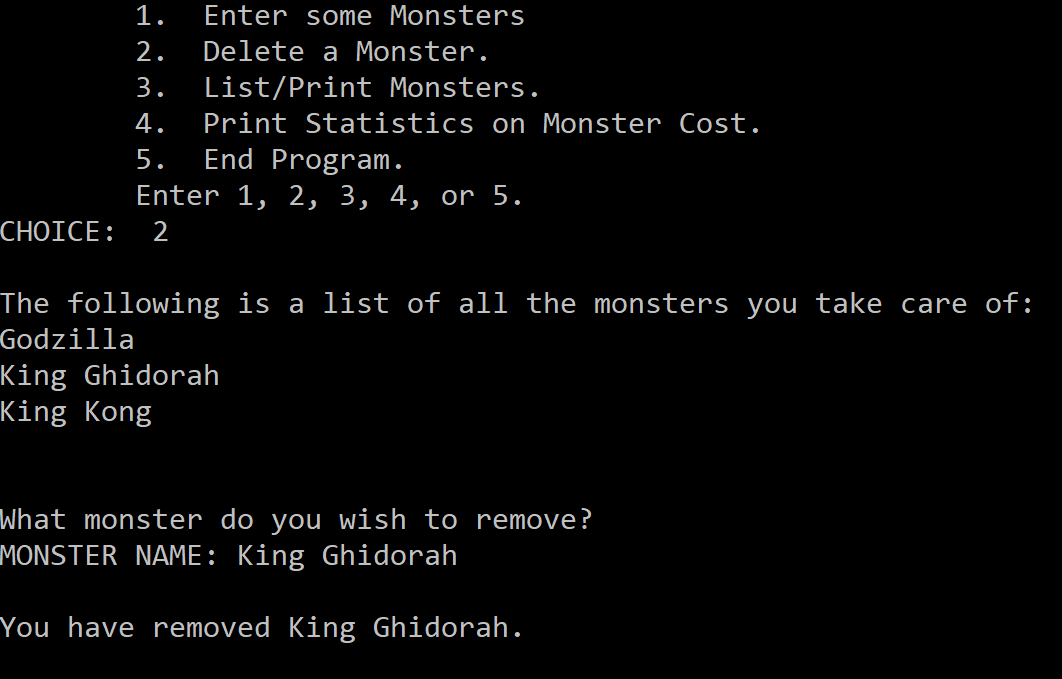
Then, increment the number of Monsters by one. Then, ask the user if they want to add another Monster. If they do, then repeat this option. If not, then just return the number of Monsters.

### deleteMonster function

The deleteMonster() function has two parameters: the current number of Monsters in the Monsters array and the Monsters array. This function returns the new number of Monsters.

First, this function will say “The following is a list of all the Monsters you take care of: “ and then it will say the name of each Monster. Then, your program will ask the user  
What Monster do you wish to remove?  
MONSTER 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 Monster. The moveArrayElements function returns a Boolean value to tell if the Monster was removed or not. Check to see if the Monster was removed. If it was, then decrement the number of Monsters and print out “You have removed [insert Monster name here].” Then return the new number of Monsters.



### moveArrayElements function

The moveArrayElements() function has the following parameters: a string with the name of the Monster that the user wishes to remove, the current number of Monsters in the Monster array, and the Monsters array. This function returns a Boolean variable that is true if the Monster was removed and false if it was not removed.

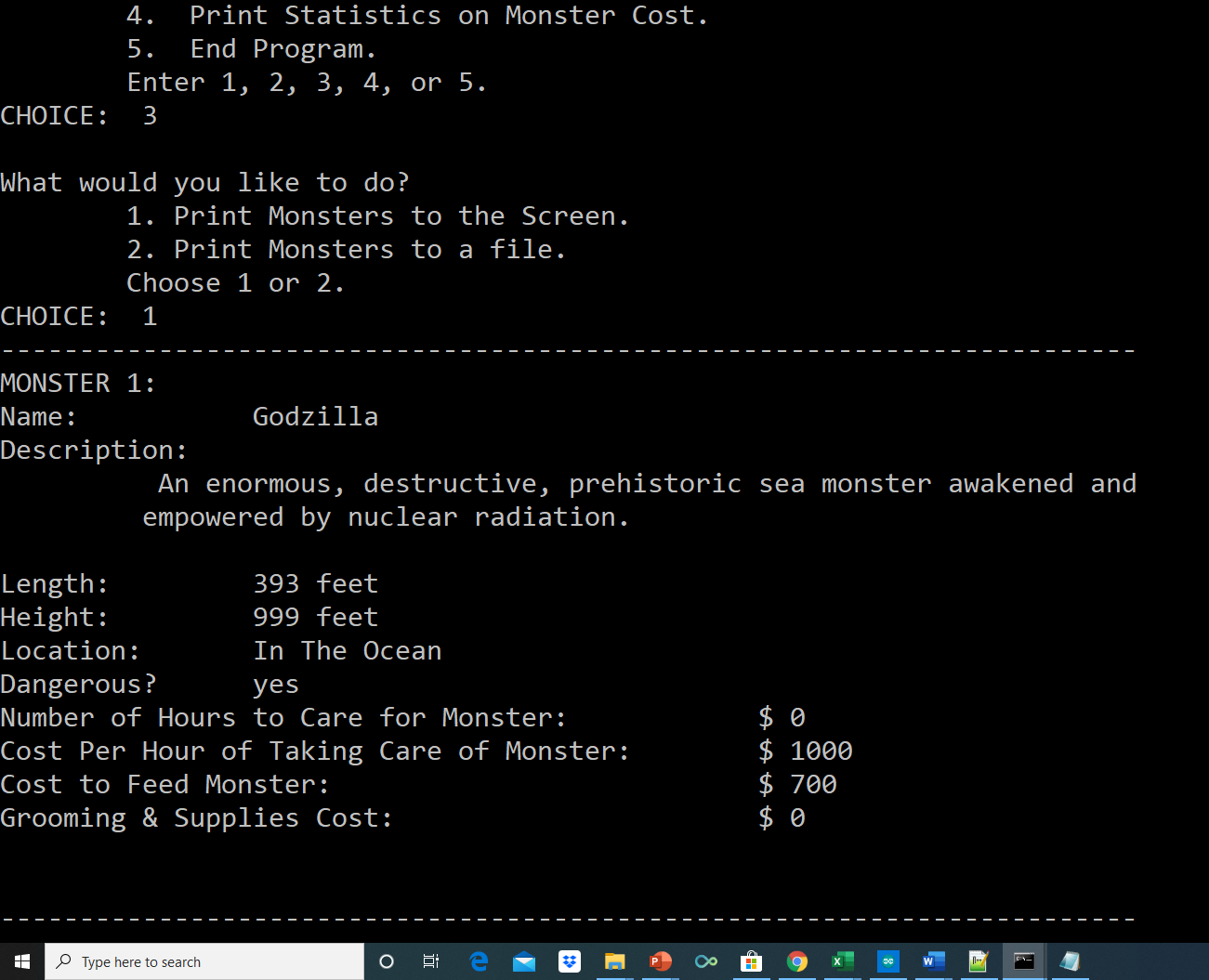
This function should first find the subscript number of the Monster that needs to be removed. Once that is found, then you know there is a Monster in the Monsters array by that name and that your program will be able to remove it. If your program cannot find the Monster in the array, then you return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

### printMonsters function

The printMonsters() function is a void function and contains the following parameters: number of Monsters currently in the Monsters array and the Monsters array. The printMonsters() function will first display a menu to the user containing the following options:

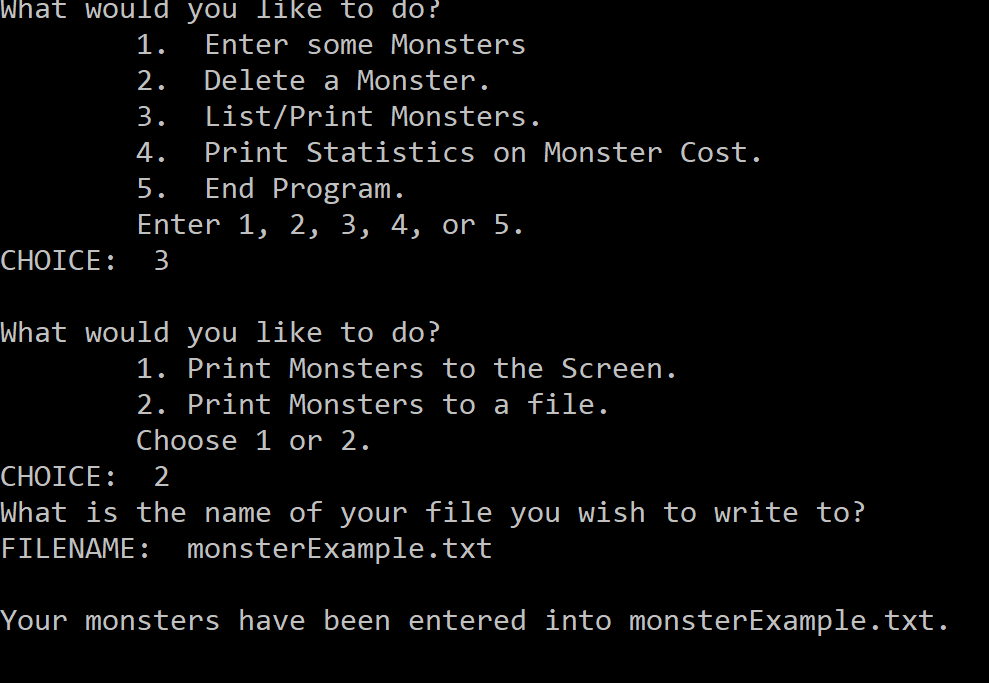
* Print Monsters to the Screen.
* Print Monsters to a File.

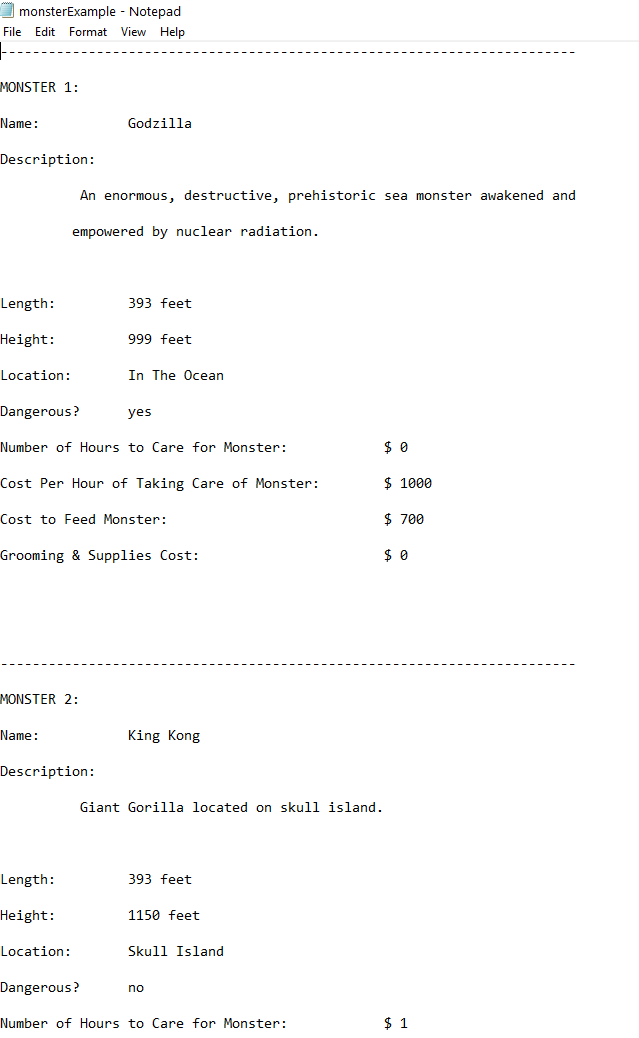
Validate the user’s choice. If the user chooses option 1, then you will print out all the Monsters in the Monsters array to the screen in the following format:



Notice that the description does word wrapping. In other words, 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 Monsters 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. Example below:





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

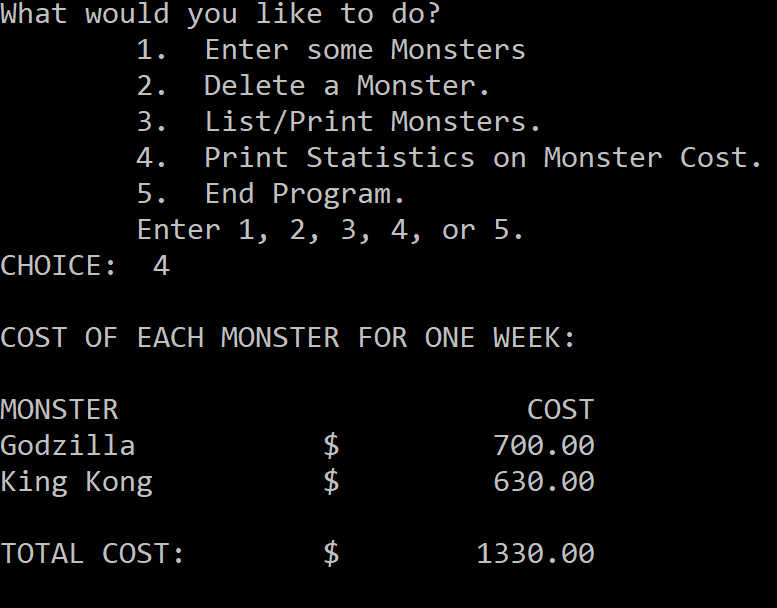
### printStatistics function

The printStatistics() function is a void function and it contains two parameters: the current number of Monsters in the Monster array and the Monsters array.

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

Cost = numHours \* costPerHour + foodCost + materialCost

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



### saveMonstersToFile function

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

* Name
* Description
* Avg. Length
* Avg. Height
* Location
* Dangerous
* numHours
* costPerHour
* foodCost
* MaterialCost

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 Monsters array to the file, this function should print out a message to standard output saying “Your Monsters were successfully saved to the [insert filename here] file.”

### convertToFloat function

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

float convertToFloat(string s)

{

istringstream i(s);

float x;

if (!(i >> x))

x = 0;

return x;

}

You will need to insert the following header file to use the convertToFloat function:

#include <sstream>