CST8219 – C++ Programming
Lab 3

Introduction:
The goal of this lab is to practice the difference between stack and heap memory.

Reference:
Git commit: https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners
Week 3 Powerpoint materials on Brightspace.

Steps:

1. Use your project from lab 2. Open git bash and navigate to your project on your hard drive. In git bash, type **git branch** to see that you are on the master branch. This means that your current project is the result of applying all the commits in time. Type the command: **git branch Week3**

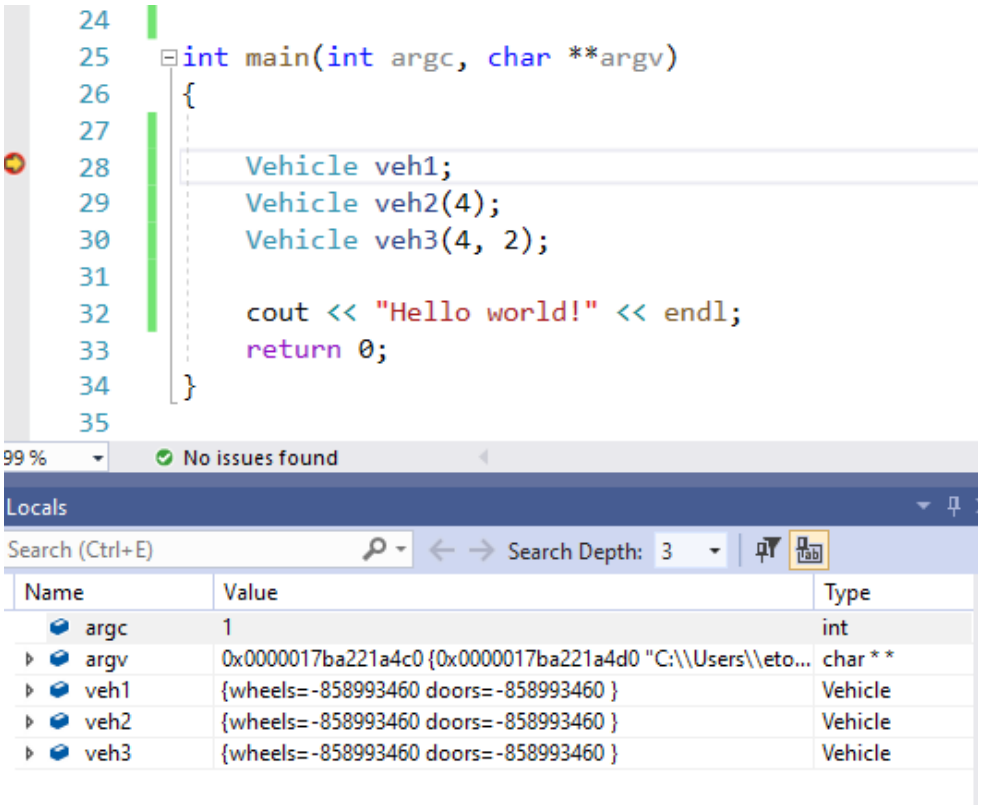
2. This makes an alternate path for building your files. Type git branch and you should now see **master**, and **Week3** in the list. There is a star next to master meaning that the next commit gets added to that path. Type **git checkout Week3** so that git will now add commits to the Week3 path instead. If you type git branch now, you'll see master and Week3 in the list, but Week3 has a star next to it meaning that any changes get put to the Week3 path instead.

```
etoru@TORUNSKI-DESKTOP MINGW64 ~/source/repos/Week3 (master)
$ git branch
* master

etoru@TORUNSKI-DESKTOP MINGW64 ~/source/repos/Week3 (master)
$ git branch week3

etoru@TORUNSKI-DESKTOP MINGW64 ~/source/repos/Week3 (master)
$ git branch
* master
  week3

etoru@TORUNSKI-DESKTOP MINGW64 ~/source/repos/Week3 (master)
$ git checkout Week3
Switched to branch 'Week3'
M       InputExample.h
M       Week3.cpp

etoru@TORUNSKI-DESKTOP MINGW64 ~/source/repos/Week3 (Week3)
$ git branch
  master
  week3
```

3. Modify your main function so that you call all 3 constructors that you had in lab 2:
   Vehicle veh1; // This calls constructor Vehicle()
   Vehicle veh2(4); //This calls constructor Vehicle(int);
   Vehicle veh3(4, 2); //This calls constructor Vehicle(int, int);

4. Add the lines to see how many bytes each variable takes:
   cout << “Vehicle takes ” << sizeof(***veh1***) << endl;
   Repeat this for veh2 and veh3.

5. Use the debugger to take screenshots of Visual Studio as you step over one line at a time to show how the stack memory changes as you progress through your code. For example:



   From the screenshot, on line 37, stack memory is allocated but has not been initialized yet. Step through each line of code and take screenshots to show the variables veh1, veh2, and veh3 get initialized, and one screenshot after veh3 is initialized (4 screenshots total). This is also showing that before calling the constructors, your variables have garbage values to begin with and they get set to normal values after the constructors.

6. Create a new variable: Vehicle *pVehicle ;

   Ask the user to input the doors and wheels to create a new Vehicle:

   cout <<”enter number of doors” << endl;
   cin >> d;
   cout << “enter number of wheels << endl;
   cin >> w ;

However use the examples from class to validate the input that the user gives. You should also check that the numbers are greater than 0 as well.
Once you have two valid numbers, create the object using **new** keyword:

pVehicle = new Vehicle(w, h);

7. Put the code from step 6 in a do / while loop asking the user if they want to create a new vehicle or quit. The user should type the char 'q' to quit, otherwise they go back and create another vehicle as in step6. Remember that if they are repeating, you will have to delete the memory allocated from the previous object. This should cause the destructor you wrote in lab 2 to output a message to the console.

8. If the user selects to quit, then the program exits the main() function and the objects veh1, veh2, and veh3 get deleted after the return 0; line, which should also output your destructor message.

9. When you are debugging, step over your lines of code until you reach the last line of the main function, which should be **return 0;**

   Take a screenshot of the local variables window in the debugger. This is all the memory that was left on the memory stack at the end. Looking at your screenshot, list the variable names, the variable Type for each, and the number of bytes each variable takes. By variable type, I mean is it an **int**, **char***, **Vehicle**, **Vehicle***, etc.

   Once you are finished, put your 5 screenshots in the same directory as your week2.cpp and CMakeLists.txt file. Use the command: git add …. to add each file to your project. Then use git bash to commit your work to the Week3 branch:
   git commit -am "Finished Week 3"
   Create a zip file containing everything in your week2 directory and submit it on Brightspace. Make sure it includes week2.cpp, CMakeLists.txt, the .git folder, as well as your 5 screenshots.

Marks: (total of 16)

You have all 3 Vehicle constructors using stack memory	+1
You have 4 screenshots of initializing stack memory	+4
The object pVehicle is created using heap memory, and deleted properly	+4
You have a loop asking the user to input the dimensions of the new object	+1
The user input is validated	+1
If the user creates another pVehicle, the previous one is deleted	+1
You have calculated the number of bytes each variable on the stack uses using the sizeof() function	+4