CSE 165: Object Oriented Programming
Spring 2022
Lab 5 (100 points)

This programming assignment has six tasks, complete each task as instructed. To submit your assignment, please organize your code in the folder "Lab5" by placing your code in it's corresponding sub-folder. For example, store your code (source, header, and assets) for task 1 in the following directory "Lab5/1/". Then, submit the compressed version of folder Lab5 to CatCourses. Submissions must arrive one minute before the lab section of week 5 (2/14 – 2/18).

1 Getters and Setters (20 points)

Create a C++ class named Circle, which you should store in a file called Circle.h. Your class should have three private variables, namely x, y, and r, all of type double. The variables x and y will store the Cartesian position of the center of the circle and the variable r should store the radius. You should also add a public variable, area, which will store the area of the circle. There should be two constructors, one that takes no arguments and instantiates a unit circle centered at the origin. The second constructor should take in three arguments, namely x, y and r, and instantiate a circle of radius r, centered at (x, y). There should be getters and setters for all the private variables. Your code will be tested with the circles.cpp file.

```
Sample output:
    Center: (0, 0)
    Radius: 1
    Area: 3.14159

    Center: (1, 2)
    Radius: 3
```

2 Inheritance in C++ (20 points)

Get the file Animal.h. It contains an Animal class that stores the name and age of an animal. Besides the appropriate constructors, getters, and setters it has a function called feed() which prints out the message "Some meal, please!!". Dogs are animals too, so we can extend the Animal class to produce a Dog class. We only need to change the constructors and destructors of the Dog class to print the appropriate messages and we need to change the feed() function to print a message saying "A small bone, please!". Your class should be stored in a file called Dog.h. Your solution will be tested with the file animals.cpp.

```
Sample output:
    Creating Generic Animal Creating Dog
    Bruno is 4 years old.
    A small bone, please!! Deleting Dog
    Deleting Generic Animal
```

3 More on Inheritance (20 points)

Suppose we wish to store several different animals in an array. We do not know ahead of time what animals they will be. C++ will allow us to create a vector of pointers to Animal, and store in that vector, any object of type Animal or a descendant of Animal. So, if we have a generic animal we can store it as a generic animal but if we have a dog we can store it as a dog.
Reuse the Animal.h and Dog.h files from previous exercises and get the display.h file. Then write a C++ program that reads in an integer N. This is followed by N lines, where each line contains a character, a string and an integer, separated by space. The character will either have the value A or D, indicating whether the animal described on this line is a generic animal or a dog. The string indicates the animal's name and the integer, its age.
For each line of input, instantiate the appropriate object and push it to the vector of pointers to Animal that you have created. Once you have pushed all the animals to the vector, call the display function, found in display.h, with your vector passed as an argument. For example, if input (marked as bold) is entered, you should see following Output:
2
A Snoopy 4
Creating Generic Animal
D Bruno 5
Creating Generic Animal Creating Dog
Snoopy
Bruno

4 Counting Objects (10 points)

Modify the Animal class used in previous exercises by adding a variable named count. This variable should store the number of instances of the Animal object that have been created in the program countingAnimals.cpp.

```
Creating Generic Animal
Creating Generic Animal
2
Deleting Generic Animal
Deleting Generic Animal
```

5 Constructors and Destructors (20 points)

Extend the file stack.h containing Stack class given in order for it to store double numbers (instead of void* pointers). Add a destructor that will delete all the stack by making calls to pop(), and for each element destroyed the element will be printed to the output. Now add two constructors: the default one that will create an empty stack, and another one that will receive an integer n and will build a stack of n elements, such that the first element is 1.0, and the next ones are incremented of 0.1. Upload your Stack class. It will be then tested for correctness with stacks.cpp.

```
2 1.5 1 0.5
1.3 1.2 1.1 1 end.
```