COMP 53: Pointers Lab, part 1

Instructions: In this lab, we are going to review pointers, allocating and deallocating memory at runtime, and memory regions.

- Get into groups of **at most two people** to accomplish this lab.
- At the top of your source code files list the group members as a comment.
- Each member of the group must individually submit the lab in Canvas.
- This lab includes **23 points** in aggregate. The details are given in the following.

1 `City` and `CoastalCity`

Extend `coastalcity.h` from the previous lab as follows:

1. Add a second constructor that receives the name, population, water name, and number of beaches as input and sets up the data components ***(2 points)***.

2 Function `main`

Include `city.h` (from previous lab) and `coastalcity.h` in `main.cpp`. In function `main()` do the following step by step:

1. Declare an object of class `CoastalCity` with the second constructor you defined above, where the name is San Diego, population is 1500000, water name is Pacific Ocean, and number of beaches is 5. Let's call this object `ccity1` ***(2 points)***.
2. Print the address of where `ccity1` resides ***(1 points)***.
3. Which memory region does this address belong to? Put your answer as a comment in the line succeeding the print statement from previous step. For example, if it resides in static memory put `//static memory` in the next line ***(1 points)***.
4. Declare a pointer to a `CoastalCity` object and nullify it. Let's call this pointer `ccityPtr1` ***(2 points)***.
5. Print the value stored in `ccityPtr1`. `ccityPtr1` is nullified, the value must be `0` ***(1 points)***.
6. Assign the address of `ccity1` to `ccityPtr1`. Next, print the value stored in `ccityPtr1`. Note that the value stored in `ccityPtr1` must be the same as the address of `ccity1` after the assignment ***(2 points)***.
7. Use `ccityPtr1` as a handle to call `printInfo()`. You can do this in two different ways:
   (a) Use `->` syntax ***(1 points)***.
   (b) Next, use `*` syntax ***(1 points)***.

   Try it in both ways!
8. Declare a pointer to a `CoastalCity` object and assign to it the address of the object created by `new` command. `new` is used to allocate memory at runtime. The `CoastalCity` object that you are creating using `new` command must have the following details: Name to be Miami, population to be 500000, water name to be Atlantic Ocean, and number of beaches to be 8 ***(3 points)***.
9. Print the address stored in `ccityPtr2` ***(1 points)***.

10. Which memory region does this address belong to? Put your answer as a comment in the line succeeding the print statement from previous step ***(1 points)***.

11. Use `ccityPtr2` as a handle to call `printInfo()`. Similar to step 7, try it in both ways ***(2 points)***.

12. Deallocate the memory to which `ccityPtr2` points to ***(1 points)***.

13. Print the address stored in `ccityPtr2` ***(1 points)***. Note that the address stored in this pointer does not change after the deallocation. However, the content of the memory in that address is not accessible anymore (try it!).

The output of the program may look like the following:

```
The address of ccity1: 0x7ffcf45be140
The address stored in ccityPtr1: 0
The address stored in ccityPtr1: 0x7ffcf45be140

ccityPtr1 used to call printInfo()
Using -> syntax:
Name: San Diego
Population: 1500000
Water: Pacific Ocean
No. of Beaches: 5

Using * syntax:
Name: San Diego
Population: 1500000
Water: Pacific Ocean
No. of Beaches: 5

The address stored in ccityPtr2: 0x215b030

ccityPtr2 used to call printInfo()
Using -> syntax:
Name: Miami
Population: 500000
Water: Atlantic Ocean
No. of Beaches: 8

Using * syntax:
Name: Miami
Population: 500000
Water: Atlantic Ocean
No. of Beaches: 8

The address stored in ccityPtr2: 0x215b030
```

Note that the addresses may be different than this, based on the memory regions that your Operating System allocates for your program.