COMP 53: Pointers Lab, part 2

Instructions: In this lab, we are going to review pointers and how to avoid memory leaks.

- 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 **27 points** in aggregate. The details are given in the following.

1 `City` and `CoastalCity`

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

1. Add a default constructor that assigns `N/A` and `0` to city's name and population ***(2 points)***.
2. Add a second constructor that receives name and population as input, and assigns them to city's name and population ***(2 points)***.

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

1. Add a new private data component which is a pointer to some `City`. Call it `adjCity`. This points to some adjacent city of a coastal city ***(2 points)***.
2. Initialize `adjCity` in the default constructor by assigning `N/A` to its name, `0` to its population ***(2 points)***.
3. Add two extra input arguments to the second constructor that demonstrate the adjacent city's name and population. Then in the body of the constructor, initialize `adjCity` with the input name and population ***(2 points)***.
4. Extend `printInfo()` function by printing the details about the adjacent city. For this purpose, invoke the `printInfo()` function of the adjacent city ***(2 points)***.
5. Define a getter function that returns the pointer to adjacent city ***(2 points)***. Call the function `getAdjCity()`.

2 Function `main`

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

1. Allocate memory for a coastal city object in the heap, and assign the address of it to a pointer. Let's call this pointer `ccityPtr`. The object must have the following details: Name to be Miami, population to be 500000, body of water name to be Atlantic ocean, number of beaches 8, and the adjacent city to be Coral Gables with population 50000 ***(4 points)***.
2. Print the address of the coastal city object from previous step ***(2 points)***.
3. Print the address of the adjacent city to the coastal city from previous step ***(2 points)***.
4. Print the information about the coastal city, by calling `printInfo()` ***(2 points)***.

The output of the program may look like the following:

```
address of coastal city: 0x1a36c20
address of adjacent city: 0x1a36c90
Name: Miami
Population: 500000
Water: Atlantic Ocean
No. of Beaches: 8
Adjacent City:
Name: Coral Gables
Population: 50000
```

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

3 Memory leak

In `main()`, if you delete `ccityPtr`, then the object will be removed from the heap. However, adjacent city data will not be removed, since it resides in another place in the heap. This causes **memory leak**. To avoid memory leak, you need to explicitly define a destructor.

1. Define the destructor for `CoastalCity`. Within the destructor delete the memory allocated for adjacent city ***(3 points)***.

Now, when you delete `ccityPtr`, destructor will be invoked, which avoids causing memory leaks.