1. The following piece of code never deallocates the objects on the heap after allocating them. Fix this program. The `deallocate` and `allocate` functions cannot be modified and they should be the only ones used to allocate and deallocate objects.

```
#include <iostream>
#include <exception>

int *a, *b, *c;

void allocate() {
   a = new int(1);
   b = new int(2);
   c = new int(3);
}

void deallocate() {
   std::cout << "Deallocating objects" << std::endl;
   delete a;
   delete b;
   delete c;
}

int main() {
  allocate();
  throw std::runtime_error(" :) ");
  deallocate();
}
```

2. The following piece of code fails in one its statements and the program finishes without reaching the end. Analyze why does it happen this behavior.

```
#include <iostream>
#include <fstream>

void openFile(std::fstream &file) {
   file.open("test.txt",std::fstream::trunc | std::fstream::app);
}

void writeToFile(std::fstream &file) {
   file << "hola" << std::endl;
}

void closeFile(std::fstream &file) {
   file.close();
}

int main() {
  std::fstream file;
  openFile(file);
  writeToFile(file);
  closeFile(file);
```

```
}
```

3. Fix the code so it actually writes the sentence “hola” in the file “test.txt”. Your fixings are however restricted to respect the following set of rules:

a) the only code you can modify is in the main function

b) the main function has to call to the functions openFile, writeToFile and closeFile in that specific order

4. Define a class for representing graphs. Be care when defining the interface of this class, we will use this class a lot during this semester. Write as many additional classes as you need to represent graphs.