Exercise 2:

Submit your solutions (source code) to the questions below by email to your instructor and TA(s) by Monday, 15th (16:30).

Question 1: An array-based stack (50 points).

In this question, we are going to implement a class Stack. We will use an array as a container for the elements of the stack. Elements of the stack will be of type 'int'.

Create the files: "Stack.h", "Stack.cpp" and "test_stack.cpp".

- "Stack.h" will contain the definition of the class Stack.
- "Stack.cpp" will contain the implementation of the methods of the class Stack.
- "test_stack.cpp" will contain the main function that will test our created class.

Step 1: Type the following code in the file "Stack.h":

```
// Stack.h
#ifndef STACK_H
#define STACK_H

class Stack {
public:
```

```
  Stack(int N) : size(0), max_size(N), top(-1), data(new int[N]) {}
  ~Stack() { delete[] data; }

  // push the element el in the stack
  // print an error message if trying to push an element in a full stack
  void push(int el);

  // pop the element on top of the stack and return it
  // print an error message if trying to pop from
  //   an empty stack and returns a dummy int
  int pop();

  // return true if stack is full
  bool is_full();

  // return true if stack is empty
  bool is_empty();

  // return the num of elements in the stack
  int num_elements();

private:
  int size;
  int max_size;
  // top is the index to the topmost element of the stack
  int top;
  int* data;
};

#endif // STACK_H
```

Step 2: Implement the methods push, pop, is_full, is_empty and num_elements in the file "Stack.cpp"

Step 3: Type the following code in the file "test_stack.cpp". You will use this code to test your stack implementation.

```
// test_stack.cpp
#include <iostream>
#include "Stack.h"

int main(void) {
  Stack s(5);

  if (s.is_empty()) std::cout << "Empty stack" << std::endl;

  s.push(2);
  s.push(5);
  s.push(7);
  s.push(9);
  s.push(9);

  if (s.is_full()) std::cout << "full stack" << std::endl;

  std::cout << "Num of elements: " << s.num_elements() << std::endl;

  int t;
  while (!s.is_empty()) {
   t = s.pop();
   std::cout << t << std::endl;
   std::cout << "Num of elements: " << s.num_elements() << std::endl;
```

```
  }
}
```

Question 2: Static members (50 points).

Write a class UniqueID such that each instance of this class has a unique identifier (the type of this identifier should be int). The identifier for the first instance of UniqueID should be one, then each new instance will have an identifier with the value one plus the identifier of the previously created object. The identifier can be accessed with the method getID().

Define the class UniqueID in files named UniqueID.h and UniqueID.cpp. To test your code create a file testUniqueID.cpp and type the code below:

```
// testUniqueID.cpp

#include <cassert> // for assert()

// COMPLETE: include the proper header

int main(void) {
  UniqueID uid1;
  assert(uid1.getID() == 1);

  UniqueID uid2;
  assert(uid2.getID() == 2);

  UniqueID uid3;
```

```
  assert(uid3.getID() == 3);

  return 0;
}
```