HW #6 3-digit numbers (polymorphism)

- In this assignment, you will implement several C++ classes to represent all natural numbers consisting of at most 3 digits (all numbers from 0 to 999).
- You will do this by filling in the data and function definitions.

- Note that in many places the keyword **&**, **const** or **virtual** was missing on purpose. Add them if you consider is necessary.
- Moreover, in this specification, should there be any type declaration error in function prototypes, fix them.

HW #6 (2)

```
// Class representing 1-digit natural numbers (from 0 to 9)
class Number {
  // Declare a protected instance variable:
  // n, 1-digit integer initialized to 0

  Number () { this->n =0; }
  Number (int n) { }
    // Check that n is a 1-digit number,
    // if not, terminate using exit ()
    // initialize instance variable to n
```

HW #6 (3)

```
Number (Number n) { } // Initialize instance variable

int getNumber ()       { } // Return the 1-digit number

bool equals (Number n) { }
  // Return true if this number is equal to n
bool compare (Number n) { }
  // Return true if this number is greater than n
std::string toString() { }
  // Return a string consisting of this number
```

HW #6 (4)

```
// Class representing 2-digit numbers (from 00 to 99)

class TwoDigitNumber: public Number {
  // Declare two protected instance variables
  // n1, Number representing first digit
  // n2, Number representing second digit

  TwoDigitNumber () { } // Initialize all digits to 0
  TwoDigitNumber (Number n1, Number n2) { }
    // Initialize all digits to n1 and n2
```

HW #6 (5)

```
bool equals (TwoDigitNumber n) { }
  // Return true if this TwoDigitNumber number
  // is equal to n
bool compare (TwoDigitNumber n) { }
  // Return true if this number is greater than n
  // Example: 21 > 11 and 10 > 08

std::string toString() { }
  // Return a string consisting of this
  // TwoDigitNumber; for example 11 or 04.
```

HW #6 (6)

```
// Class representing 3-digit numbers (from 000 to 999)

class ThreeDigitNumber: public TwoDigitNumber {
  // Declare two protected instance variables
  // n1, Number representing first digit
  // n2, TwoDigitNumber representing next 2-digits

  ThreeDigitNumber (Number n1,
                    TwoDigitNumber n2) { }
    // Initialize all digits to n1 and n2
```

HW #6 (7)

```
bool equals (ThreeDigitNumbers n) { }
  // Return true if this ThreeDigitNumber number
  // is equal to n
bool compare (ThreeDigitNumber n) { }
  // Return true if this number is greater than n

std::string toString() { }
  // Return a string consisting of this
  // ThreeDigitNumber, for example 111 or 003.
```

HW #6 (8)

Other tasks:

- Write a class **TestNumbers** that implements the following four functions:

  1. **Number** getFirstDigit (int n)
  2. **Number** getSecondDigit (int n)

  that return a **Number** corresponding to the first and the second digit of a 2-digit natural number n, respectively.

- For example, if n = 75, then

  getFirstDigit (n) returns the **Number** object corresponding to number 7 and getSecondDigit (n) returns the **Number** object corresponding to number 5.

HW #6 (9)

- You may want to implement other similar functions if needed.

- Implement a function

  3. static **Number** *genNums () [100]

  that returns an array of (pointers to) 100 random numbers of 1-digit, 2-digit, or 3-digit, namely from 0 to 999.

- **Hint:** use of the concept of polymorphism.

HW #6 (10)

- Implement a function

4. static std::string printAvg (**ThreeDigitNumber** *nums[ ])

   that returns the string representation of the **average** from an array of (pointers to) ThreeDigitNumber numbers.

- **Hint:** you are required to traverse and examine the numbers generated from 3. genNums (), taking only those ThreeDigitNumbers to serve as your input.

- Use the above functions to test your program in **main**.