CMPUT 350 Lab 2 Exercise Problems

Rules:

- You can use all course material and man pages, but no other information such as web pages, books, or written notes. Using other information sources during the exercise constitues cheating.
- Your programs must compile without warning using

  ```
  g++ -Wall -Wextra -Wconversion -Wsign-conversion -O -g -std=c++17 ...
  ```

  In case there are compiler warnings or errors, marks will be deducted.
- Test your programs with different values. For now, the speed of your program is irrelevant. So don't spend time on optimization
- You must check for the appropriate preconditions/postconditions. Your program shouldn't crash or have undefined behaviour (**hint**: use asserts)!
- Your programs must be well structured and documented. Use ctrl-x t in Emacs to pretty-print it. Marks are assigned to functionality, program appearance, and comments.
- In case your program hangs, use ctrl-c to terminate it.
- Remember that you need to include the appropriate header files. To find out which ones you need for specific functions such as printf, use then man command.

Submit your solution files `Matrix.cpp matrixTest.cpp` on eClass under "Lab 2 / Submission".

Important: Submit often (the system will only accept solutions submitted before 16:50)

---

1. [25 marks] Consider the following Matrix class that represents an r×c int matrix, and stores all elements in a one-dimensional array listed below.

Using the provided `Matrix.h`, create files `Matrix.cpp`, and `matrixTest.cpp` in which you implement its methods, and test them, respectively.

`Matrix.h` must not contain any method implementation!

Sample `print()` output looks like this (r=3, c=4):

```
0 1 2 3
4 5 6 7
8 9 10 11
```

```
// Matrix.h
class Matrix {
public:
    // construct r by c matrix and initialize elements with 0
    // pre-condition: _r > 0, _c > 0
    Matrix(int _r, int _c);

    // release all resources
    ~Matrix();

    // CC
    Matrix(const Matrix &rhs);

    // AO
    // pre-condition: rhs must have same shape
    Matrix &operator=(const Matrix &rhs);

    // return size (number of total elements)
    int size() const;

    // set matrix element (r,c) to value
    // (indices start with 0)
    // pre-condition: r,c within range
    void set(int r, int c, int value);

    // return matrix element (r,c)
    // (indices start with 0)
    // pre-condition: r,c within range
    int get(int r, int c) const;

    // print all elements to stdout - row by row, using space as separator
    void print() const;

    // return true iff m has the same size (rows and cols) as maxtrix and identical elements
    bool equals(const Matrix &m) const;

private:
    // data
    int r;      // rows
    int c;      // cols
    int *p;     // sole owner of r*c ints
};
```