Applied Programming
Submission 4

In this assignment you are going to extend what you did in the last assignment, such that the internal representation of the matrix no longer is an array, but instead uses a one-dimensional std::vector. In addition you also have to make it possible to define whether the new internal representation should be row or column major. Specifically the following should be **added** to your existing Matrix.hpp code:

- A new private variable: *bool isRowMajor*
- A new public method: *std::vector<T> getData() const*, which should return the vector
- A new public method: *bool getOrder() const*, which should return the isRowMajor variable

The following should be **changed** in your existing Matrix.hpp code:

- The array representing the matrix should be changed to: *std::vector<T> mData*, which later must be initialised to a one-dimensional vector (not two-dimensional)
- The constructor should be changed to: *Matrix(int numRows, int numCols, bool rowMajor)*, and you should initialize the new private variable *isRowMajor* with *rowMajor*. The mData vector should be initialized as a zeroed one-dimensional vector (not two-dimensional)

You can of course create additional functions as you want to make the needed changes to the existing code in order to facilitate the new internal representation, but how the existing functions are called must not be changed. See page 3 for what part of your header file should look like.

The handin format is the same as usual.

Part 2

In the second part of the assignment we are going to implement a k-Nearest-Neighbour Algorithm(kNN). Given a set of points and a query point, the kNN returns the k points with the smallest distance to the query point, ordered by distance.

In this assignment, we are training the use of iterators, another core idiom of C++. Your assignment is to complete the functions specified in the assignment4-2.hpp. You are allowed to add new functionality as well as to use the whole standard library. For this assignment, the headers iterator vector and algorithm will be very helpful.

What to hand in

As in the last assignment you should only handin a .zip file containing the two header files of the matrix and vector class., i.e. Matrix.hpp and Vector.hpp. For the second part of the assignment you also need to upload assignment4-2.hpp, Vector_old.cpp, and Vector_old.hpp. A total of 5 files.

How to get started

Before trying to submit to the code checker you should get the following to work, which test the added/changed parts of the code.

```
#include "Matrix.hpp"

int main(int argc, char const *argv[])
{

    Matrix<int> m(2,2,0); // column major
    m(2,1) = 1337;

    Matrix<int> m2(2,2,1); // row major
    m2(2,1) = 1337;

    assert(m.getOrder() == 0);
    assert(m2.getOrder() == 1);

    assert(m.getData()[1] == m(2,1));
    assert(m(2,1) == 1337);
    assert(m.getData()[1] == 1337);

    assert(m2.getData()[2] == m2(2,1));
    assert(m2(2,1) == 1337);
    assert(m2.getData()[2] == 1337);

    return 0;
}
```

Part of modified header file

```
#include "vector"
template <typename T>
class Matrix
{
private:
    int mNumRows, mNumCols; // dimensions

    std::vector<T> mData;

    bool isRowMajor;
public:
    Matrix(const Matrix<T>& otherMatrix);
    Matrix(int numRows, int numCols, bool rowMajor);
    ~Matrix();
    int GetNumberOfRows() const;
    int GetNumberOfColumns() const;
    T& operator()(int i, int j); //1-based indexing
    //overloaded assignment operator
    Matrix<T>& operator=(const Matrix<T>& otherMatrix);
    Matrix<T> operator+() const; // unary +
    Matrix<T> operator-() const; // unary -
    Matrix<T> operator+(const Matrix<T>& m1) const; // binary +
    Matrix<T> operator-(const Matrix<T>& m1) const; // binary -
    // scalar multiplication
    Matrix<T> operator*(T a) const;
    double CalculateDeterminant() const;
    // declare vector multiplication friendship

    template <typename u>
    friend Vector<u> operator*(const Matrix<u>& m,
                               const Vector<u>& v);
    template <typename u>
    friend Vector<u> operator*(const Vector<u>& v,
                               const Matrix<u>& m);

    // new public functions in Matrix.hpp
    std::vector<T> getData() const;
    bool getOrder() const;

};
```