Applied Programming
Submission 2

The format is the same is in assignment 1, but this time you must name your submission handin2.

What to hand in

Exercise: **6.1.1-7**, **6.2**, **7.1**, **8.2**, **9.1**.

Exercise 6.1.1-6

Provided header file: *ComplexNumber.hpp*.

```
#ifndef COMPLEXNUMBERHEADERDEF
#define COMPLEXNUMBERHEADERDEF

#include <iostream>

class ComplexNumber
{
    private:
        double mRealPart;
        double mImaginaryPart;
    public:
        ComplexNumber();
        ComplexNumber(double x, double y);
        double CalculateModulus() const;
        double CalculateArgument() const;
        ComplexNumber CalculatePower(double n) const;
        ComplexNumber& operator=(const ComplexNumber& z);
        ComplexNumber operator-() const;
        ComplexNumber operator+(const ComplexNumber& z) const;
        ComplexNumber operator-(const ComplexNumber& z) const;
        friend std::ostream& operator<<(std::ostream& output, const ComplexNumber&
   z);

        //exercise prototypes
        double GetRealPart() const;
        double GetImaginaryPart() const;
        friend double RealPart(const ComplexNumber& z);
        friend double ImaginaryPart(const ComplexNumber& z);
        ComplexNumber(const ComplexNumber& z);
        ComplexNumber(double real);
        ComplexNumber CalculateConjugate() const;
        void SetConjugate();
        //not mandatory, but useful for exercise 6.1.7
        ComplexNumber operator*(const ComplexNumber& z) const;

};

#endif
```

Exercise 6.1.7

Provided header file: *CalculateExponential.hpp*.

Write a function that given a complex matrix A and a sufficiently high value *nMax* calculates the exponential of A, and stores the result in *res*.

Note that the formula for computing the exponential of a matrix is missing an exclamation mark. The denominator in the summation should be n!(as seen in equation 2 on `http://mathworld.wolfram.com/MatrixExponential.html`)

```
#ifndef _CALCULATEEXPONENTIAL_
#define _CALCULATEEXPONENTIAL_

#include "ComplexNumber.hpp"

void CalculateExponential(ComplexNumber **A, int nMax, ComplexNumber **res);

#endif
```

Exercise 6.2

Provided header file: *Matrix2x2.hpp*.

Remember to also implement the get methods for the private variables.

```
#ifndef SUBMISSION_MATRIX2X2_HPP_
#define SUBMISSION_MATRIX2X2_HPP_

#include <iostream>

class Matrix2x2 {
    private:
        double val00; // first row, first column
        double val01; // first row, second column
        double val10; // second row, first colum
        double val11; // second row, second column

    public:
        Matrix2x2();
        Matrix2x2(const Matrix2x2& other);
        Matrix2x2(double a, double b, double c, double d);
        double CalcDeterminant() const;
        Matrix2x2 CalcInverse() const;

        Matrix2x2& operator=(const Matrix2x2& z);
        Matrix2x2 operator-() const;
        Matrix2x2 operator+(const Matrix2x2& z) const;
        Matrix2x2 operator-(const Matrix2x2& z) const;

        void MultScalar(double scalar);

        double Getval00() const {return val00;}
        double Getval01() const {return val01;}
        double Getval10() const {return val10;}
        double Getval11() const {return val11;}
};
#endif /* SUBMISSION_MATRIX2X2_HPP_ */
```

Exercise 7.1

Provided header files: *Student.hpp*, *GraduateStudent.hpp*, *PhdStudent.hpp*.

The variables and methods follow the naming suggested in the exercise text. The variable defining if a GraduateStudent is part or full time is here defined as a boolean: it is 1 if the GraduateStudent is full time and 0 otherwise.
Note that a GraduateStudent only owes library fines, and that PhDStudent owes nothing.

```
//Student.hpp
#ifndef SUBMISSION_STUDENT_HPP_
#define SUBMISSION_STUDENT_HPP_

#include <string>
#include <iostream>

class Student {
public:
    Student();
    Student(std::string name, double fines, double fees);

    std::string name;
    double tuition_fees;
    virtual double MoneyOwed() const;

    void SetLibraryFines(double amount);
    double GetLibraryFines() const;

private:
    double library_fines;
};
#endif /* SUBMISSION_STUDENT_HPP_ */


// GraduateStudent.hpp
#ifndef SUBMISSION_GRADUATESTUDENT_HPP_
#define SUBMISSION_GRADUATESTUDENT_HPP_

#include "Student.hpp"

class GraduateStudent : public Student {
private:

public:
    GraduateStudent();
    GraduateStudent(std::string name, double fines, double fees, bool fullTime);
    bool fullTime;
    virtual double MoneyOwed() const;

};
#endif /* SUBMISSION_GRADUATESTUDENT_HPP_ */


// PhdStudent.hpp
#ifndef SUBMISSION_PHDSTUDENT_HPP_
#define SUBMISSION_PHDSTUDENT_HPP_

#include "GraduateStudent.hpp"

class PhdStudent : public GraduateStudent {
public:
    PhdStudent(std::string name, double fines, double fees, bool fullTime);
    virtual double MoneyOwed() const;
};
#endif /* SUBMISSION_PHDSTUDENT_HPP_ */
```

Exercise 8.2

Provided header file: *Exercise82.hpp*.

You should **not** make a corresponding *.cpp* file for this header, but just fill in the body of the function defined in the header file. If you want to know why you should not make a *.cpp* file, you can read about it here `https://isocpp.org/wiki/faq/templates#templates-defn-vs-decl`.

```
#ifndef SUBMISSION_EXERCISE82_HPP_
#define SUBMISSION_EXERCISE82_HPP_

template<typename T>
T CalcAbs(T val) {
    //fill this out yourself
}
#endif /* SUBMISSION_EXERCISE82_HPP_ */
```

Exercise 9.1

Provided header files: *Exception.hpp*, *OutOfRangeException.hpp*, *FileNotOpenException.hpp*.

```
// Exception.hpp
#ifndef EXCEPTIONDEF
#define EXCEPTIONDEF

#include <string>
class Exception {
    private:
        std::string mTag, mProblem;
    public:
        Exception(std::string tagString, std::string probString);
        void PrintDebug() const;
};

#endif //EXCEPTIONDEF
```

```
// OutOfRangeException.hpp
#ifndef SUBMISSION_OUTOFRANGEEXCEPTION_HPP_
#define SUBMISSION_OUTOFRANGEEXCEPTION_HPP_

#include "Exception.hpp"

class OutOfRangeException : public Exception {
    public:
        OutOfRangeException(std::string prob);
};
#endif /* SUBMISSION_OUTOFRANGEEXCEPTION_HPP_ */


// FileNotOpenException.hpp
#ifndef SUBMISSION_FILENOTOPENEXCEPTION_HPP_
#define SUBMISSION_FILENOTOPENEXCEPTION_HPP_

#include "Exception.hpp"

class FileNotOpenException : public Exception {
    public:
        FileNotOpenException(std::string prob);
};
#endif /* SUBMISSION_FILENOTOPENEXCEPTION_HPP_ */
```