Assignment 13 - Streams, Multiple Inheritance and Exceptions

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules:
  https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 13.1 *Copy a file* (1 point)

Presence assignment, due by 11:00 AM today **Graded manually**
Language: C++

Write a program which reads from the standard input the name of a file (e.g., `"input.txt"`). Your program should use C++ streams to create a copy of this file with the name as the original file name concatenated with `"_copy"` and then the original extension (e.g., `"input_copy.txt"`).
You can assume that the input will be valid and the necessary input file has a valid content if existing.

Problem 13.2 *`Complex` class with streams* (2 points)

Due by Monday, December 2nd, 23:00 **Graded manually**
Language: C++

Adapt and extend your previous source code for working with complex numbers (`Complex.h`, `Complex.cpp` and `testComplex.cpp`) such that the operators + for adding, − for subtracting, * for multiplying two complex numbers, = for assigning, << and >> for input and output streams are overloaded. Your testing program (`testComplex.cpp`) should read two complex numbers from the files called `in1.txt` and `in2.txt`, then compute their sum, the difference and the product, and print the results on the screen as well as into a file called `output.txt`. You have the freedom to set the structure of the input files.
You can assume that the input of the testing program will be valid and the necessary input files have a valid content if existing.

Problem 13.3 *Concatenate `n` files into new file* (2 points)

Due by Monday, December 2nd, 23:00 **Graded manually**
Language: C++

Write a program which reads from the standard input an integer value `n`, followed by `n` file names. Your program has to concatenate the content of the `n` files and write the result into the file called "`concatn.txt`" using binary read and write. Add a `'\n'` to separate the contents of the different files inside the resulting file. The operation of the concatenation is successful only if all files are existing and their opening was successful.
You can assume that the input will be valid and the necessary input files have a valid content if existing.

Problem 13.4 *Multiple inheritance I* (1 point)

Due by Monday, December 2nd, 23:00 **Graded manually**
Language: C++

Consider the following source file:
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/minheritance1.cpp`
Compile and run the file. If you find any compilation errors, explain their reason as comments in the code. Then fix the errors such that the program compiles and runs. Explain the motivation of your modifications in the code and their effects on the execution.

Problem 13.5 *Multiple inheritance II* (1 point)

Due by Monday, December 2nd, 23:00 **Graded manually**

Language: C++

Consider the following source file:

`https://grader.eecs.jacobs-university.de/courses/320142/cpp/minheritance2.cpp`

Compile and run the file. If you find any compilation errors, explain their reason as comments in the code. Then fix the errors such that the program compiles and runs. Explain the motivation of your modifications in the code and their effects on the execution.

Bonus Problem 13.6 *`Matrix` and `Vector` classes* (3 points)

Due by Monday, December 2nd, 23:00 **Graded manually**

Language: C++

Extend your previous source code for working with vectors (`Vector.h` and `Vector.cpp`) and write a `Matrix` class (`Matrix.h`, `Matrix.cpp`) for operations with matrices. If you did not write a `Vector` class for one of the previous bonus problems, then write one with minimal functionality as needed for this problem. Then write a testing program which illustrates operations between matrices and vectors (`testmatvec.cpp`) using the operations described as in the following.

- Overload the operators << and >> to be able to enter matrices and vectors from the standard input and from files (e.g., `in1.txt`, `in2.txt`, etc.), and to send matrices and vectors to the standard output and to files (e.g., `out1.txt`, `out2.txt`, etc.). You have the freedom to set the structure of the input files.
- Overload the operator * for computing the product of a vector and a matrix, and a matrix and a vector. For simplicity reasons you can consider a vector to be either a row vector or a column vector such that mathematical multiplication with a matrix is possible. By this convention, the result of the multiplication is also a vector (row vector or column vector).

Your implementation has to check if the operations between matrix and vector, and vector and matrix are valid or not (concerning the compatibility between a vector and matrix concerning the amount of elements in them). Besides this you can assume that the input of the testing program will be valid, the necessary input files have a valid content if existing.

Problem 13.7 *Out of range exception* (1 point)

Due by Monday, December 2nd, 23:00 **Graded manually**

Language: C++

Write a program which creates a vector of integers and adds the value 8, 20 times into the vector. Include the library `<vector>` and then create a `vector` object which stores the 20 values from above. Then write a `try` and `catch` block in which your code should try to access the 21^{th} element from the vector using the `at()` method. The exception you should catch should be of type `out_of_range`. In the `catch` block use `cerr` to print to the standard error stream the type of the exception by calling the redefined `what()` method inherited from the `exception` class.

Bonus Problem 13.8 *Simple different exceptions* (2 points)

Due by Monday, December 2nd, 23:00 **Graded manually**

Language: C++

Write a program and a function with an integer parameter which can throw four exception types: a `char`, an `int`, a `bool`, and your own exception class called `OwnException` derived from `exception`. If the parameter of the function is 1 then the character `'a'` should be thrown, if it's 2 then the integer `12` should be thrown, if it's 3 then the `bool` value `true` should be thrown, and in the default case an `OwnException` with the string `"Default case exception"` should be thrown. You should overwrite the `what()` method for the `OwnException` class by returning the string `"OwnException\n"`. Call the function in the `main` function in its four versions and catch the corresponding exceptions. In the `catch` blocks you should print to the standard error stream `cerr` the string `"Caught in main: "` followed by the value of the thrown exception. In the case of `OwnException` print the string returned by the `what()` method.

Bonus Problem 13.9 *Car exceptions* (1 point)

Due by Monday, December 2nd, 23:00 **Graded manually**

Language: C++

Write a `Garage` class that has a `Car` (i.e., object of a second class) that is having troubles with its `Motor` (i.e., object of a third class). Use a function-level try block in the `Garage` class constructor to catch an exception (thrown from the `Motor` class with the string "`This motor has problems`") when its `Car` object is initialized. Throw a different exception with the string "`The car in this garage has problems with the motor`" from the body of the `Garage` constructor's handler and catch it in the `main` function.

How to submit your solutions

- Your source code should be properly indented and compile with `gcc` or `g++` depending on the problem without any errors or warnings (You can use `gcc -Wall -o program program.c` or `g++ -Wall -o program program.cpp`). Insert suitable comments (not on every line ...) to explain what your program does.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   CH-230-A
   a13_p1.[c or cpp or h]
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://cantaloupe.eecs.jacobs-university.de`.
  If there are problems (but **only** then) you can submit the programs by sending mail to `k.lipskoch@jacobs-university.de` **with a subject line that begins with CH-230-A**.
  It is important that you do begin your subject with the coursenumber, otherwise I might have problems to identify your submission.
- Please note, that after the deadline it will not be possible to submit any solutions. It is useless to send late solutions by mail, because they will not be accepted.

This assignment is due by Monday, December 2nd, 23:00.