Assignment 5 - Exceptions and Blackjack

- The problems of this assignment must be solved in C++.
- The TAs are grading solutions to the problems according to the following criteria: https://grader.eecs.jacobs-university.de/courses/320143/2019_1r3/Grading-Criteria-C++.pdf

Problem 5.1 *Out of range exception* (1 point)

Presence assignment, due by 18:30 h today

Write a program which creates a `vector` (i.e., STL container) of characters and adds the character '`@`', 15 times into the vector. Then write a `try` and `catch` block in which your code should try to access the 16^{th} element (i.e., the element at position 15) 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 `what()` method inherited from the `exception` class.

Problem 5.2 *Simple different exceptions* (2 points)

Presence assignment, due by 18:30 h today

Write a program and a function with a character parameter which can throw four exception types: a `char`, an `int`, a `bool`, and your own exception class called `OwnException` derived from the `exception` class. If the parameter of the function is '1' then the character '`e`' should be thrown, if it's '2' then the integer `99` should be thrown, if it's '3' then the `bool` value `false` should be thrown, and in the default case an `OwnException` with the string
"`This is a default case exception`" should be thrown. You should overwrite the `what()` method for the `OwnException` class by returning the string
"`This is an OwnException`". 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 "`Exception caught in main: `" followed by the value of the thrown exception. In the case of an `OwnException` print the string returned by the `what()` method.

Problem 5.3 *Car exceptions* (1 point)

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.

Problem 5.4 *Reorganize the code* (1 point)

Download
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/blackjack.cpp`
Consider the slides from Lecture 5 & 6 (pages $3-9$) as additional explanation.
Reorganize the code such that each class is put into its own header file and the corresponding implementation into its own `cpp` file. Then put the `main` function into another `cpp` file. You will possibly need to include several other classes for several files. Do not use `std` namespace in header files, but change the code accordingly (e.g., `std::cout`). You might also need guards via conditional compilation in your header files.
You can use the following makefile for compiling your source files:
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/MakefileBJ`.
You can assume that the input will be valid. Submit all source files as a zip archive.

Problem 5.5 *Implement two methods* (1 point)

Implement the method `Card::GetValue()`. If the face is down, then the value is considered to be 0, otherwise the values is the face of the numbered cards, and 10 for Jacks, Queens or Kings. Also implement the method `Hand::GetTotal()`. By default consider an ace to count 1 point and if the total is low enough then count 11 points.
You can assume that the input will be valid. Submit all source files as a zip archive.

Problem 5.6 *Implement method* (1 point)

Implement the method `Deck::Deal()`.
You can assume that the input will be valid. Submit all source files as a zip archive.

How to submit your solutions

- Your source code should be properly indented and compile with `g++` without any warnings (You can use `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.
  Each program **must** include a comment on the top like the following:

```
/*
   CH08-320143
   a5_p1.cpp
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://grader.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 CH08-320143**.
  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 Tuesday, April 16th, 10:00 AM.