

Custom Exception Class Exercises

- When we write our own exception class, why is it better to derive from a class in the `std::exception` hierarchy than to write a completely new class?
- Which standard header should we include when deriving from a class in the `std::exception` hierarchy?

- When we write our own exception class, which member functions should we consider implementing?
- In the `bad_student_grade` class, why did we not provide any data members?
- Why did we not implement a copy constructor or destructor for the class?

- Write a program which reads a number from standard input and creates an instance of StudentGrade
- Your program should handle any exceptions which are thrown during this process
- Test your program, entering both valid and invalid grades
- (The source code for the bad_student_grade and the StudentGrade classes is provided on the following pages)

bad_student_grade class

```
class bad_student_grade : public std::out_of_range {  
public:  
    // Default constructor  
    bad_student_grade() : std::out_of_range("Invalid grade: please try again") {}  
    // We need a constructor which takes a string, for consistency with std::exception  
    bad_student_grade(const string& s) : std::out_of_range(s) {}  
    bad_student_grade(const char *s) : std::out_of_range(s) {}  
    // The default copy constructor is good enough as we do not have any data members  
    // bad_student_grade(const bad_student_grade& other) : std::out_of_range(other) {}  
  
    // Finally, we need to override the virtual what() member function  
    const char* what() const noexcept { return out_of_range::what(); }  
};
```

StudentGrade class

```
class StudentGrade {  
    int result;  
    public:  
        StudentGrade(int grade) : result(grade) {  
            if (grade < 0 || grade > 100) {  
                throw bad_student_grade();  
            }  
        }  
};
```