You shall submit a zipped, **and only zipped**, archive of your homework directory, hw2. The directory shall contain, at a minimum, the files `hw2/parse_scores.cc` and `hw2/parse_scores.h`. Name the archive submission file hw2.zip

I will use my own makefile and test files to compile and link to your `hw2/parse_scores.cc` and `hw2/parse_scores.h` files. You must submit, at least, these two files.

Now that you have learned to use functions and arrays in C++, you are able build a C++ API library. The functions must accept arrays of strings which provided by functional unit tests in the directory.

These functions will parse an arrays of strings into meaningful values. An array of strings will have the following format:

{ student_id, number_of_grades, grade_0, grade_1, grade_n }

Such that a possible array would be:

{ "flast", "3", "93.2", "88.1", "97.6" }

Read the provided header file documentation for instructions on how the functions should behave. For more details, you must open and read the provided tests. Students asking questions which can be answered by reading the tests will be referred to the tests.

Notes:

Global variables should be avoided because this is an exercise in functional programming. Global variables are often a lazy way of avoiding classes, which are not needed for this task.

Conversion from strings to numeric values can be accomplished with functions such as:

- **std::stoi** and
- **std::stod**.

Their documentation can be found on cplusplus.com

Errors occur in three ways:

1. **Truncation** errors: these are detected by counting:
   - Any correct input array must have at least two elements: **student_id** and **number_of_grades**.
   - The array size given to a function *will be correct*. This implies that your array might specify more grades than actually exist. Consider the following parameters:
     ({ "gsmith", "3", "89.0", "77.1" }, 4)
     The error can be detected when **number_of_grades** is parsed and converted. There is no way for a third grade to exist because the total array size is 4.
2. **Validation** errors: The format of the data is as follows and any deviation should be considered a format error.
   - number_of_students: non-negative integer
   - student_ids: non-negative integer

- number_of_student grades: non-negative integer
- student_grade: non-negative floating point (use double precision)

3. **Corruption** errors: sometimes the values get corrupted. Though there is no way of knowing whether an ID is corrupt (because all IDS of length one or more are valid) the numerics can be detected. Consider the parameters:
   ({ "gsmith", "/3", "8*.0", "77dot1" }, 4)
   If using `std::stoi` or `std::stod`, you can check for these errors by catching the exceptions those functions generate:

```
#include <stdexcept>  // std::invalid_argument
#include <string>  // std::stoi, std::stod
.
.
.
try {
  student_no = std::stoi(input[index]);
} catch (const std::invalid_argument& exception) {
  return false;  // non-numeric student number
}
```

Testing and Points

You are provided a basic test app which should be used to ensure that your code is somewhat correct. I would suggest a more rigorous testing scheme. Your code must behave as indicated in the documentation. Passing the tests is not enough to guarantee full credit.
The point breakdown is as follows:

- Compilation: 0.25 points per test, 1.0 in total (15% penalty)
- GetId: 1 point
- GetGrade: 2.5 points
- GetGradeCount: 1 point
- GetMaxGrade: 2.5 points
- Style & Documentation: 2 points

Late assignments will lose 10% per day late, with no assignment begin accepted after 3 days (at 70% reduction in points).

Check your syllabus for the breakdown of grading.

The End.