Exercise 14:

Submit your solutions (source code) to the questions below by email to your instructor and TA(s) by Monday, January 28th (16:30).

Question 1: Practice with some common algorithms (30 points).

This question will make you practice with some of the algorithms provided by the STL.

Please save your answers to the questions below in a file named "test_various_algorithms.cpp". Using algorithms provided by the STL do the following:

- Remove all characters from a string that are not alphabetic. Use: remove_if defined in the header <algorithm>, string::erase and isalpha declared in the header <cctype>
- Given a string, transform all its characters to lowercase (i.e. "Hello" would be transformed to "hello"). You should overwrite the input string. Use: transform from <algorithm> and tolower from <cctype>

Question 2: Palindromes (35 points).

Using the solution of question 1, please write a function named isPalindrome(). This function takes an input string and returns a bool equal to true if the input string is a palindrome. A palindrome is a word or a phrase that can be read in forward and backward direction. Please ignore spaces, punctuation and capitalization when checking if an input string is a palindrome.

```
// palindrome.cpp
#include <string>
#include <iostream>
#include <cassert>

using namespace std;

// COMPLETE: define isPalindrome(string& input);

int main(void) {
  string palindrome1 = "Don't nod";
  string palindrome2 = "A Toyota! Race fast... safe car: a Toyota";
  string not_palindrome = "Random string";

  assert(isPalindrome(palindrome1)==true);
  assert(isPalindrome(palindrome2)==true);
  assert(isPalindrome(not_palindrome)==false);

  cout << "Tests passed" << endl;

  return 0;
}
```

Question 3: More algorithms (35 points).

Write functions compute_average() and compute_standard_deviation() that take as argument a range of iterators over doubles and return a double corresponding to the average and standard deviation of the input data. Please write these functions using accumulate() (in the header <numeric>) to compute the sum and distance() (in the header <iterator>) to compute the number of elements in the range of iterators.

```
#include <vector>
#include <cmath>
#include <cassert>

using namespace std;

int main(void) {
 vector<double> input;
 for (int i = 0; i < 50; ++i) {
  input.push_back(double(i) / 10.0);
 }

 double avg = compute_average(input.begin(), input.end());
 double stddev = compute_standard_deviation(input.begin(), input.end());

 assert(fabs(avg - 2.45) < 1e-5);
 assert(fabs(stddev - 1.44309) < 1e-5);

 return 0;
}
```