NIU CSCI 340 Assignment

Vector, random numbers and reverse copy.

In this assignment, the student will write several C++ functions, to support a program that generates random integers, store them in a STL vector, and reverse copy the contents to another vector, then print them out. You are provided a main() function and a makefile for this assignment for your local testing. (Note that not all assignments will provide main() and makefile. You may be expected to write your own main() and makefile in some cases.)

This assignment will be submitted via the autograder. You will implement the required functions in the file *solutions.cc,* and submit that file to the autograder. You should feel free to create whatever files you want to test things locally, including writing your own simple programs to test small parts on their own, but your submission should only include the required code.

Specifications

You need to implement the following functions:

1. void gen_rnd_nums (vector < int >& v , int seed, int range_limit, int vec_size);

This function generates vec_size integers and puts them in the vector v.

Do not assume the vector has been pre-allocated enough space. Do not assume the vector passed in was empty. So you need to clear the contents first, and then use push_back() to add the generated numbers. Or you can resize the vector before you assign values.

Because this is an early assignment for the semester, you are not required to use iterators to work with the vector (but you can do so if you prefer).

You will generate random integers in the range of [LOW = 1, HIGH = range_limit]. You need to initialize the random number generator by calling the function *srand ( )* with the provided seed value. You only need to seed the random number generator once. You can then generate random integers by calling the function *rand( )* multiple times. The function *rand()* gives a random number between 0 and RAND_MAX (a very large number defined by the system), inclusive. You need to figure out how to get a random number in the range of [1, range_limit] inclusive. Hint: use modulus.

2. int reverse_copy(vector < int >& v1, vector < int >& v2);

Check the size of the two vectors and return -1 if not the same.
Then copy the contents of v1 into v2, in reserve order.

3. void print_vec (const vector <int>& v, int items_per_row, int item_width);

This function displays the contents of vector v on *stdout*, printing exactly items_per_row on a single line, except perhaps the last line.

The numbers need to be properly aligned on the output. For each printed number, allocate the width of each element on *stdout* based on the argument item_width. *See std::setw for details.*

Print a newline after you print the entire vector. (Note that the auto-grader may be strict when comparing the outputs you need to examine the format in addition to the contents.)

Other Requirements

You must add a documentation box at the top of the .cc file, in the following format:

```
/*****************************************************************************
* NIU CSCI 340 Section YOUR_SECTION *
* Assignment #     ← based on the assignment # assigned to this assessment by your instructor
* YOUR FULL NAME - YOUR Z-ID *
* *
* I certify that everything I am submitting is either provided by the professor for use in *
* the assignment, or work done by me personally. I understand that if I am caught submitting *
* the work of others (including StackOverflow or ChatGPT) as my own is an act of Academic *
* Misconduct and will be punished as such. *
* *
* * A brief description of the program should also be added here.
*****************************************************************************/
```

How To Submit

To submit this program, you will upload the required files to the NIU Autograder on Blackboard.

Grading Considerations

- Does it compile? Does it run?
- Does the output match? The output from your implementation will be tested against the output from the reference in the auto-grading system.
- Did you use the data structure required you to use if any?
- Did you do the needed error checking for your functions if required?
- Did you document your code?
    1. You need a docbox at the top of every one of the files you're submitting as explained above. You should also add a description of what the program does.
    2. You should add a docbox or line comments for every function that you implement, explaining what it does and what each parameter is for. Be consistent: If you use docbox, then do so for all functions.
    3. Add other comments inside your code blocks describing what you're doing and why.
    4. The use of doxygen style comments is encouraged, but not required.