Name: _____

Solutions

1. (1 pt.)
   - **Read all material carefully.**
   - *If in doubt whether something is allowed, ask, don't assume.*
   - You may refer to your books, papers, and notes during this test.
   - E-books may be used *subject to the restrictions* noted in class.
   - Computers are not permitted, except when used strictly as e-books or for viewing ones own notes.
   - Network access of any kind (cell, voice, text, data, ...) is not permitted.
   - Write, and draw, carefully. Ambiguous or cryptic answers receive zero credit.
   - Use class and textbook conventions for notation, algorithmic options, etc.
   - Do not attach or remove any pages.

   Write your name in the space provided above.

   Do not write on this page below this point.

2. (2 pts.) Provide a single C++ statement that defines a C++ STL *vector*, named `v5`, of five unsigned integers and initializes it to contain the elements (in index order): 3, 1, 4, 1, 5.

   (A) `vector<unsigned int> v5 {3, 1, 4, 1, 5};`

3. (2 pts.) Provide a single C++ statement that prints, to *standard output*, the number of elements (items) in a C++ STL *vector* named `howMany`, whose elements are of type `float`.

   (A) `std::cout << howMany.size();`

4. (2 pts.) Provide a single C++ statement that declares a C++ STL *vector*, named `hislah`, containing three elements of type `char`, and initializes it to contain the elements (characters, in index order): `y`, `e`, and `s`.

   (A) `vector<char> hislah {'y', 'e', 's'};`

5. (2 pts.) Provide a single C++ statement that adds the character `!` (exclamation mark) as the fourth element of the vector of Question 4.

   (A) `hislah.push_back('!');`

6. (2 pts.) Provide a single C++ statement that reverses (undoes) the change made by Question 5 (that is, removes the fourth element added there).

   (A) `hislah.pop_back();`

7. (14 pts.) Provide **well-formatted source code of a complete C++ program** that

   (a) Defines a function `rot_vec` that takes a single vector (of arbitrary length) as argument and that cyclically rotates its elements to the left by one position (so that the new item at index 0 is the one previously at index 1, the new one at index 1 is the one previously at index 2, etc., and the new one at the last position is the one originally at index 0).

   (b) Uses the above function in the `main` function with a suitably defined vector to illustrate its operation by printing the vector's elements before and after the function is invoked.

   Poorly formatted, messy, or otherwise hard to read code will result in very substantial loss of points. *Explain your answer briefly, especially to qualify for partial credit.*

   Ⓐ *Note: This answer is much, much more detailed than needed because it incorporates some suggestions/questions raised in class.*

```
#include <vector>  // needed for using STL vector
#include <iostream> // needed for cout etc.

using namespace std; // convenience, else std::cout etc.

/*
  Cyclic-rotates left the elements of v.

  In response to a Q in class: It is very important to have the &
  below so that v is a reference argument instead of the default
  (without &) which is a value argument.  In the latter case, what
     the
  function would get would be a copy of the vector from the code that
  calls it, so all its modifications would be made on that copy and
  not on the original one as needed.
 */
void rot_vec(vector<int> & v) {
  int len = v.size();
  if (len > 1) { // else nothing to be done
    int v0 = v[0]; // save v[0] and then assign each element its next
        one
    for(int i = 0; i < len - 1; i++) {
      v[i] = v[i+1];
    }
    v[len - 1] = v0; // set last element to saved v0
  }
}

/*
  To illustrate the importance of the & noted above, here is an
     almost
  identical version of the function, the only difference being the
  name and the lack of &.  As the code below illustrates, it does not
  change anything in the vector given to it by code that calls it.
```

```
 */
void rot_vec_noop(vector<int> v) {
  int len = v.size();
  if (len > 1) { // else nothing to be done
    int v0 = v[0]; // save v[0] and then assign each element its next
        one
    for(int i = 0; i < len - 1; i++) {
      v[i] = v[i+1];
    }
    v[len - 1] = v0; // set last element to saved v0
  }
}

int main() {
  vector<int> d = {3, 1, 4, 1, 5, 9}; // sample vector
  for(auto elem : d) cout << elem << " "; // initial state
  cout << endl;
  rot_vec(d); // rotate it
  for(auto elem : d) cout << elem << " ";  // changed state
  cout << endl;
  rot_vec(d); // rotate it again
  for(auto elem : d) cout << elem << " ";  // changed state
  cout << endl;
  rot_vec_noop(d); // does not change d; a no-op in that sense.
  for(auto elem : d) cout << elem << " ";  // changed state
  cout << endl;
  return 0;
}
```