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

cout << someVec.size();

3. ✓ (2 pts.) Provide a single C++ statement that prints, to *standard output*, the **number of bytes** used by C++ STL *vector* named `someVec`, whose elements are of type `float`.

cout << sizeof(someVec);

4. ✓ (2 pts.) Provide a single C++ statement that prints, to *standard output*, the **number of elements** (items) in an array named `someArr`, whose elements are of type `float`.

cout << sizeof(someArr)/sizeof(float);

5. ✓ (2 pts.) Provide a single C++ statement that prints, to *standard output*, the **number of bytes** used by an array named `someArr`, whose elements are of type `float`.

cout << sizeof(someArr);

6. ✓ (2 pts.) Provide a single C++ statement that defines an *array*, named `aNums`, of five unsigned integers and initializes it to contain the elements (in index order): 3, 1, 4, 1, 5.

unsigned int aNums[5] = {3, 1, 4, 1, 5}

7. ✓ (2 pts.) Provide a single C++ statement that defines a C++ STL *vector*, named `vNums`, of three unsigned integers and initializes it to contain the elements (in index order): 2, 3, 5.

vector<unsigned int> vNums {2, 3, 5}

8. ✓ (17 pts.) Provide **well-formatted source code of a complete C++ program** that

(a) Defines the array aNums as in Question 6.

(b) Defines the vector vNums as in Question 7.

(c) Prints the elements of aNums on *standard output* on a single newline-terminated line, with a single space after each element.

(d) Prints the elements of vNums as above.

(e) Swaps second element (that is, the element at index 1) of aNums with the second element of vNums (so that the new second element of aNums is the old second element of vNums, and vice versa).

(f) Extends vNums to contain five numbers (instead of the original three), with the two new elements, in index order, being the corresponding elements of aNums.

(g) Prints the (current) elements of aNums as done earlier.

(h) Prints the (current) elements of vNums as done earlier.

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

necessary headers for cout & vector type-obj

yeah namespace

maine function

declare our array & vector

loop aNums to print & add the \n

same thing, vectors are nice

spare variable buf holds data for swap

push_back the last two elements with corresponding aNums

print again

```
#include <vector>
#include <iostream>
using namespace std;

int main() {
      unsigned int aNums[5] = {3, 1, 4, 1, 5} ✓
      vector<unsigned int> vNums {2, 3, 5} ✓
      for (auto a : aNums) cout << a << ' '; ✓
      cout << endl; ✓
      for (auto v : vNums) cout << v << ' ';
      cout << endl;
      int buf = aNums[1]; ✓
      aNums[1] = vNums[1]; ✓
      vNums[1] = buf; ✓
      vNums.push_back(aNums[3]); ✓
      vNums.push_back(aNums[4]); ✓
      for (auto a : aNums) cout << a << ' '; ✓
      cout << endl;
```

continues on next page

[additional space for earlier material]

```
    for (auto v : vNums) cout << v << ' ';
    cout << endl;
    return 0;
}
```

✓

// would have been more OOP but no time

OK!

9. ✓ (15 pts.) Provide **well-formatted source code of a complete C++ program** that

(a) Defines a function vec_zero_some that sets *some specified* elements of a given vector of ints to zero. The elements to be set to zero are specified by an array of ints, whose elements are the *indices* of the vector that are to be set to zero. In more detail, the function takes three arguments, vec, arr, and n that are, respectively, the vector of ints that is to be modified, the array of indices of vec (that are to be zeroed), and the number of elements in arr. Invoking (executing) the function should result in all elements of vec that are at an index position that occurs in arr being set to zero.

(b) Demonstrates the operation of this function using a suitably defined vector and array, both of whose elements are printed 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 for better partial credit.*

headers for cerr & vectors

ditto #8.

& alias to modify

old C for loop to avoid & op. shenanigans, set specified indices to 0

maine function

vector declaration don't question taco

array declaration and n var

loop print
print ~~strategy~~ unspecified, cerr to be safe

call vec-zero-sum

loop print again

nice

```
#include <iostream>
#include <vector>

using namespace std;

void vec-zero-sum(vector<int>&vec, int arr[], int n) {   ✓ ✓ ✓
    for (int i; i< n; i++) {
        vec[arr[i]] = 0;   ✓
    }

int main {
    vector<int> taco {1, 1, 1, 1, 329, 1}
    int zeroes_len = 2
    int zeroes[zeroes_len]= {1, 3}
    for (auto el : taco) cerr << el;   ✓  el taco, I get it!

    vec-zero-sum(taco, zeroes, zeroes_len);

    for (auto ele: taco) cerr << ele;
    return 0;
}                                         ✓
// why is the back page not blank   ?!
```