— 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`.

std::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`.

std::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`.

std::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`.

std::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};

1~~6~~7

— 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.*

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

using namespace std;

int main() {
a    unsigned int aNums[5] = {3, 1, 4, 1, 5}; ✓
b    vector<unsigned int> vNums {2, 3, 5}; //as in 6 and 7
c    for (auto i ; aNums) cout<< i << " ";
     cout << endl; //using for loop to output elements of aNum
d    for (auto i : vNums) cout << i << " ";
     cout << endl; //using for loop to output elements of vNum
e    unsigned int t = aNums[1]; // temp variable to store aNums[1]
     aNums[1] = vNums[1]; //shuffling numbers around
     vNums[1] = t; ✓
f    for (int i = 3; i < 5; i++) vNums.push_back(aNums[i]);
```

[additional space for earlier material]

```
for (auto i: aNums) cout << i << " ";
cout << endl;
for (auto i: vNums) cout << i << " ";
cout << endl; // outputting again.
return 0;
}
```

✓

See comments;

first, variables are defined as in 6 and 7.

Then a for loop with cout to print the values

Then uses a temp variable to store aNums[1], reassigns aNums[1] to vNums[1], and sets vNums[1] to the temp variable.

Then uses push_back() member function to add indices 3 and 4 of aNums to vNums.

Finally, prints them again.

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 `int`s to zero. The elements to be set to zero are specified by an array of `int`s, 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 `int`s 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.*

```
#include<iostream>
#include <vector>
using namespace std;
void vec_zero_some(vector<int> & vec, int arr[],
int n){
  for(int i=0; i<n; i++){
    vec[arr[i]] = 0;
  }
}
int main(){
  vector<int> pi {3,1,4,1,5};
  int ind[3] = {1,2,3};
  int n=3;

 for (int i: pi)cout<< i << " "; cout<<endl;
 for (int i: ind)cout<<i<< " "; cout<< endl;
 vec_zero_some(pi, ind, n);
 for(int i: pi) cout<<i<<" "; cout<<endl;
 for(int i: ind) cout<<i<<" "; cout<<endl;
 return 0;
}
```

vec needs to be passed as reference to change it ✓

i from 0 to n-1; these are the valid indices of arr[] ✓

the ith index of arr determines which index of vec is set to 0. ✓

This will result in a compiler warning, since it looks like cout<<endl; is part of the for loops, but it isn't. It's done here to save space. ✓