Started on	Monday, November 15, 2021, 12:59 AM
State	Finished
Completed on	Monday, November 15, 2021, 1:29 AM
Time taken	30 mins

Question **1**

Complete

Points out of 5.00

Sometimes what you want to do to a collection is more complicated than just change all the 2s in here to 7s. Sometimes you might want to do something to every element in a particular range or over an entire collection. Perhaps putting the result back where it was, so take each element and double it. Perhaps putting it into a different collection, take each element in this collection and double it, putting the answer in this other, unrelated collection.

```
30  // Assume the following definition:
31  // std::vector<int> src { /* lots of interesting int initializers */ };
32  // constexpr int bigger { /* some int initializer */ };
33  // Write a single statement that uses a standard library algorithm and a lambda to
34  // scale each value in container src by the value bigger
35  std::transform(std::begin(src),std::end(src),std::begin(src),[](int x){return x*bigger;});
36  print(src);
37  std::transform(std::begin(src),std::end(src),std::begin(dest),[](int x){return x/smaller;});
38
39  // Assume the following defintiion:
40  // constexpr int smaller { /* some int initialzier */ };
41  // Now, write a single statement that uses a standard library algorithm and a lambda to
42  // scale down each value in container src by a factor equivalent to the value smaller and
43  // to write the result in container dest
44
```

Question **2**

Complete

Points out of 5.00

Hopefully in the near future, you're at a C++ job interview. Suppose the lead interviewer asks you to code on the whiteboard and someone may ask you to reverse a string in place. What would happen if you were to answer "*by calling `reverse()` on the string*." It might backfire if you've an old-school interviewer who wants to see you doing pointer arithmetic. However, in real life, using standard algorithm *std::reverse* is the simplest way to reverse a string in place.

In case you've not done the in-place-reversal of a string on a whiteboard and you get asked in an interview, this should be your answer: you get a pointer *first* that points to the first character and *last* that points to the last character; swap the characters that these pointers point to; increment the *first* pointer and decrement the *last* pointer; stop if these two pointers have crossed; otherwise continue swapping.

Since you've shown your knowledge of C++ standard algorithms, what would you do if the old-school interviewer asks you to implement your in-place-reversal of a string using a standard algorithm ***other than*** `reverse()`?

```
20  // Assume the following definition:
21  // std::string sentence { /* some string literal here */ };
22  // Write one statement that uses a standard library algorithm to reverse the characters in sentence
23  std::reverse(std::begin(sentence),std::end(sentence));
24  std::cout << sentence << '\n';
25  std::cout << "Before: ";
26  print(vi);
27  std::iter_swap(vi.begin(),vi.end()-1);
28  // Assume the following definition (in the code section above that is hidden from you)
29  // std::vector<int> vi { /* list of int initializers */ };
30  // Write a single statement that uses a standard library algorithm to swap the first and last elements in vi
31
32  std::cout << "After:  "; print(vi);
33  std::reverse_copy(std::begin(sentence),std::end(sentence),std::back_insert_iterator<std::string>(destination))
34
35  // Assume the following definition:
36  // std::string destination; // destination has zero size
37  // Write a single statement that uses a standard library algorithm to copy the characters from
38  // string sentence into destination in reverse order
39
```

Question **3**

Complete

Points out of 5.00

Say you want to find all the negative numbers in a collection and change them to 0. Perhaps it's not a collection of integers, it's a collection of some kind of an object, and you want to find all of the objects that have an outstanding balance that's negative and change their outstanding balance to 0. You could do that in a series of complicated ways, calling `find()` in some sort of loop and then reaching out through the iterator you got back from `find()`, but there is a better algorithm in the standard library.

Follow the instructions in the comment block.

```
21  // Assume the following definitions:
22  // std::vector<int> vi { /* lots of initializers of type int */ };
23  // constexpr int new_value { /* some int initializer */ };
24  // constexpr int value_to_replace { /* some int initializer */ };
25  // Write a single statement that uses a standard library algorithm to replace every occurrence of
26  // a value equivalent to value_to_replace with the value in new_value
27  std::replace(vi.begin(),vi.end(),value_to_replace,new_value);
28  print(vi);
29  std::replace_if(vi.begin(),vi.end(),[](int v){return v<minimum_value;},new_value);
30  // Assume the following definition:
31  // constexpr int minimum_value { /* some int initializer */ };
32  // Write a single statement that uses a standard library algorithm and a lambda to replace
33  // all values less than minimum_value with the value in new_value
34
```

Question **4**

Complete

Points out of 10.00

Follow the instructions in the comments block.

```
19  // Assume the following definition:
20  // std::vector<int> vi(10);
21  // Write a single statement that uses a standard algorithm and a lambda to fill vi with
22  // the following sequence of 10 values in vi: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024
23  std::generate(vi.begin(),vi.end(),[i=0]() mutable {return 2<<i++;});
24  print(vi);
25
26  // Write another single statement that uses a standard algorithm and a lambda to replace
27  // elements of vi with the following sequence of 10 values: 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
28  std::generate(vi.begin(),vi.end(),[i=10]() mutable {return --i;});
29  print(vi);
30
31  // Assume the following definition:
32  // constexpr int how_many { /* some int initializer between 1 and 10 */ };
33  // Write a single statement that uses a standard algorithm and a lambda to fill the first "how_many"
34  // number of elements of container vi with values that double every time starting with 6.
35  // Suppose how_many is initialized to 7. Then, the sequence of values in vi would be:
36  // 6, 12, 24, 48, 96, 192, 384, 2, 1, 0
37  std::generate_n(vi.begin(),how_many,[x=3]()mutable{return x*=2;});
38
```

Question **5**

Complete

Points out of 5.00

Use the following definition of class **`Resource`** and follow the instructions in the comment block.

```
class Resource {
 static inline int Count = 0;
 static void PrintCount() {
   std::cout << "Resource count: " << Count << '\n';
 }
```

```
private:
  int value{};
public:
 Resource(int val = 1) : value(val) { ++Count; PrintCount(); }
 Resource(Resource const& r) : value{ r.value } { ++Count; PrintCount(); }
 Resource& operator=(Resource const& r) = default;
 ~Resource() { --Count; PrintCount(); }
 int Value() const { return value; }
};
```

```
std::ostream& operator<<(std::ostream& os, Resource const& resource) {
 os << "{" << resource.Value() << "}";
 return os;
}
```

```
38  // Study class Resource defined earlier in the question's text.
39  // Assume the following definitions:
40  // std::vector<Resource> vr { /* some int initializers here */ };
41  // constexpr int filter_value { /* some int initializer here */ };
42  // Write a single statement that uses a standard library algorithm and a lambda to
43  // remove all resources with value filter_value. Your statement must initialize a
44  // variable end_location with an iterator to the element that follows the last element not removed.
45  auto end_location=std::remove_if(vr.begin(),vr.end(),[=](Resource& r){return r.Value()==filter_value;});
```

Question **6**

Complete

Points out of 5.00

A common programming pattern that programmers implement repeatedly is apply an operation sequentially to the elements of two containers and store the result in a third container.

```
38  // Assume the following definitions:
39  // standard container src1 with random access iterator capability containing int values.
40  // standard container src2 with bidirectional access iterator capability containing int values - assume
41  // that src2 has a size at least as large as src1
42  // an empty standard container dest with bidirectional access iterator capability
43  // Write a single statement that uses a standard library algorithm and a lambda to divide an element
44  // of src1 with a corresponding element in src2 and to store the result at the front of dest
45  std::transform(src1.begin(),src1.end(),src2.begin(),std::front_inserter(dest),[](int const& x, int const& y){retur
46
```

return x/y

Question **7**

Complete

Points out of 5.00

Follow the instructions in the comment block.

```
28  // Assume the following definitions:
29  // struct Person { std::string name{}; int id{0}; };
30  // std::vector<Person> people { /* lots of initializers of type Person */ };
31  // std::vector<Person> even_people; // empty container
32
33  // Now, use a standard algorithm and a lambda to write a single statement that copies
34  // elements with even ID values in container people to container even_people
35  std::copy_if(people.begin(),people.end(),std::inserter(even_people,even_people.begin()),[](auto& person){ if (pers
```

if(person.id%2==0){return true;}else{return false;}

return (person.id%2==0);

Question **8**

Complete

Points out of 5.00

A common programming pattern that programmers implement repeatedly is apply an operation sequentially to the elements of two containers and store the result back in one of these two containers.

```
26  // Assume a standard container src with random access iterator capability that contains values of type int
27  // Assume an empty standard container dst with bidirectional iterator capability
28  // Write a single statement that uses a standard algorithm with a lambda to take the
29  // difference between two adjacent elements in src and insert that value in the front of container dst
30  std::transform(src.rbegin(),src.rend()-1,std::next(src.rbegin()),std::back_inserter(dst),[](int lhs, int rhs){retu
```

return lhs-rhs;

Question **9**

Complete

Points out of 5.00

Use standard library algorithms and lambdas to solve the problems indicated in the comments.

```
 9      // Assume following definition:
10      // std::vector<int> vi{ /* container initialized with some ints */ };
11      // Insert one statement that uses a standard algorithm and lambda expression to initialize
12      // variable even_total (that you must define) with sum of even values in container vi:
13      int even_total=std::accumulate(vi.begin(),vi.end(),0,[](int ref,int i){return ref+=(i%2==0)?i:0;});
14      std::cout << "The summation total of even numbers is: " << even_total << '\n';
15
16      // Insert one statement that uses a standard algorithm and lambda to initialize
17      // variable odd_total (that you must define) with sum of odd values in container vi:
18      int odd_total=std::accumulate(vi.begin(),vi.end(),0,[](int ref,int i){return ref+=(i%2!=0)?i:0;});
19      std::cout << "The summation total of odd numbers is: " << odd_total << '\n';
20
21      // Insert one statement that uses a standard algorithm and lambda to initialize
22      // variable total (that you must define) with sum of values of all elements in container vi:
23      int total=std::accumulate(vi.begin(),vi.end(),0,[](int ref,int i){return ref+=i;});
24      std::cout << "The summation total of the collection of a is: " << total << '\n';
25      int product=std::accumulate(vi.begin(),vi.end(),vi.front(),[](int ref,int i){return ref*i;});
26      std::cout << ( (total != even_total+odd_total) ? "BAD\n" : "GOOD\n");
27
28      // Insert one statement that uses a standard algorithm and a lambda to initialize variable product
29      // (that you must define) with multiplicative product of values of all elements in container vi:
30
```

Question **10**

Complete

Points out of 5.00

Use standard library algorithms and lambdas to generate the necessary output. The specs are described in the code fragments.

```
ollowing definition:
tor<std::string> words{ "I"s, "love"s, "lambdas"s }; // notice trailing s ...
ne statement here that uses a standard library algorithm to
ze object all_words with the concatenation of all elements in words
 all_words=std::accumulate(words.begin(),words.end(),std::string());
< "(" << all_words.size() << ") " << all_words << '\n';

ollowing definition:
ing const pg_says{ "Prasanna Says:"s }; // notice trailing s ...
ne statement here that uses string pg_says and container words to initialize
t more_words with the following string:
a Says: I love lambdas" (double-quotes not included!!!)
 more_words=std::accumulate(words.begin(),words.end(),pg_says,[&](const std::string& w,const std::string& word){ret
< "(" << more_words.size() << ") " << more_words << '\n';
 vi_to_prefix=std::accumulate(vi.begin(),vi.end(),prefix,[&](const std::string& w, int i){return w+ " " + std::to_s

ollowing definitions:
tor<int> const vi { 1, 2, 3, 4, 5 };
ing const      prefix { "vi to string:"s }; // notice trailing s ...
ne statement here that uses the above definitions to initialize an object
efix with string "vi to string: 1 2 3 4 5" (double-quotes not included!!!)
```

Don't forget to remove the second .size()

return w + " " + word;

return w + " " std::to_string(word);

Question **11**

Complete

Points out of 5.00

Follow the instructions in the code comments.

```
// Assume following definitions:
// std::list<int> li { /* lots of int initializers */ };
// int const target_val { /* an int initializer goes here */ };

// Write a single statement that uses a standard algorithm to initialize a variable (call it what you like)
// with the location of first element in li equivalent to value target_val
auto temp=std::find(li.begin(),li.end(),target_val);

// Write another single statement that uses a standard algorithm and a lambda to zero-out
// all values in li starting at location returned by previous statement
std::for_each(temp,li.end(),[](int& i ){i=0;});
print(li);
// Write another single statement that sets all values in container li to value 2
std::for_each(li.begin(),li.end(),[](int& i ){i=2;});
```