Hint

1. ALL file containing comments: **Will Replace by OJ, DO NOT EDIT!**, will be replaced by OJ system.
2. STR token in given **tokenizer** was trimmed.
   May have some whitespaces between **key**, **=**, and **"** for an **attribute**. Given **tokenizer** will ignore them.
   There may have some **back-slash (\\)** in value of attribute.
3. TL; DR, Using the given Tokenizer will properly handle all formatting issues.

006-Sequence Transform

Difficulty: Medium   Expect Time: 45 min   Author: Aren, Guo

Please implement 3 **template functions**:

```
1.  Container map(Container, U (*functor)(T) )
2.  Container pick(Container, bool (*functor)(T) )
3.  unsigned int count(container<T>, bool (*functor)(T) )
```

They have something in common: the first parameter receives a **sequence container instance**, and the second parameter receives a **callable object (`functor`)**.

The sequence containers mentioned above, does not include **std::array,** and **callable object(functor)** refers to function pointer, function object, std::function, or lambda.

- **map** template function
  Receives a container and a functor, and returns a container with all the return values of functor. If the new container wraps a different type from the original container, user can manually specify the first template parameter.

  For example:

```
int addOne(int a) { return a + 1; }
char toAlphabet(int a) { return a + 65;  }

std::vector<int> vec = { 1,2,3,4,5 };

// return std::vector<int> { 2,3,4,5,6 }
map(vec, addOne);

// return std::vector<char> { 'B','C','D','E','F' }
map<char>(vec, toAlphabet);
```

- **pick** template function
  Receives a container and a functor, returns a container of the same type as the input container.
  If the value returned by functor is true, the element (value) is retained; otherwise, it is discarded. The new container wraps the same type as the original container.

  For Example:

```
int isEven(int a) { return a % 2 == 0;  }
int allFalse(int a) { return false;  }
std::vector<int> vec = { 1,2,3,4,5,6,7,8,9,10 };
pick(vec, isEven); // return std::vector<int> { 2,4,6,8,10 }
pick(vec, allFalse); // return std::vector<int> {}
```

- **count** template function

  Receives a container and a functor, returns a unsigned int values. This function returns the number of times the value which functor returns true.

  For example:

```
int isEven(int a) { return a % 2 == 0; }
int allFalse(int a) { return false; }
std::vector<int> vec = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
count(vec, isEven); // return 5
count(vec, allFalse); // return 0
```

  Must be implemented as a template function, because the parameters passed in different types. You need to redeclare the parameter types and return types based on the **YOUR** implementation.

Input

1. Please implement these three template functions in solution.h.
   a) Redeclare the return types and parameter types according to **YOUR** implementation based on the example provided.
   b) Considering template specialization, your implementation might exceed 3 functions.
2. The input and output will be handled by the provided code.
3. The Online Judge will replace the following files:
   a) main.cpp
4. The following files are sample test cases for Online Judge. Please copy the contents of the following file into main.cpp for testing.
   a) case1.h
   b) case2.h
5. Input range
   A. Input containers will only be **std::vector**, **std::deque**, and **std::list.**
      a) Container will not wrap pointer types, reference types, void types, enum, and union.
      b) E.g., std::vector<int*>, std::vector<int&>, and std::vector<void>
      are not to be the input.
   B. **Callable object (functor)** includes the following types: function pointer, lambda, and std::function.
      a) All functors only receive one parameter, the type of which is the same as the type wrapped by the container.
         i. For example: std::vector<int> wrap int type, so functor's parameter is int type.
      b) The functor used by map template function returns container wrap any type. If the type returned by the functor is different from the type wrapped by the container, then set the type in the template first parameters.
      c) The functor used by pick, count return bool value.

Output

1. Please DON’T print any data to STDOUT.
2. The output will be handled by the provided code
3. The sample output and the corresponding input files are shown as follow:
   a) out001.txt corresponds to case1.h
   b) out002.txt corresponds to case2.h