管理域

#A. Unit Test: STL and Templates





■ 客观题

NOTE:

该比赛已结束,您无法在比赛模式下递交该题目。您可以点击"在题库中打开"以普通模式查看和递交本题。

Unit Test: STL and Templates

Answer the following questions according to the C++17 standard.

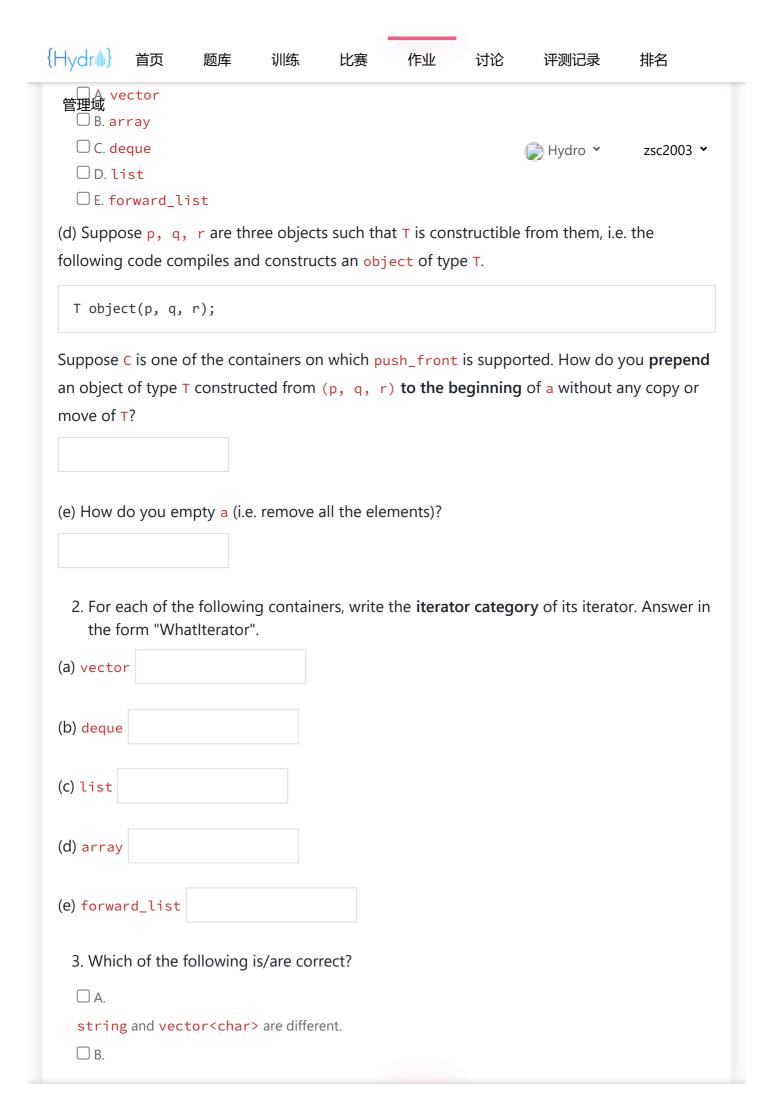
For all the questions, suppose we have the following using declarations.

```
using std::vector;
using std::list;
using std::array;
using std::deque;
using std::forward_list;
using std::string;
```

Part 1: Basics

- 1. Suppose a is of type C<T>, where C ∈ { vector , list , forward_list , deque } and T is some type that makes the type C<T> well-formed. Let x be an object of type T. For each of the following blanks, write the simplest solution. It may be either an expression or a statement.
- (a) Suppose $C \in \{ \text{ vector }, \text{ list }, \text{ deque } \}$. How do you append a copy of x to the end of a?

(b) Suppose $\mathtt{C} \in \{ ext{ vector }, \mathtt{list }, dequ \}$	}. How do you ap	ppend \mathbf{x} to the e	nd of <mark>a</mark> through a
move?			





is converted by the compiler to

```
► Hydro × zsc2003 ×
```

```
for (std::size_t i = 0; i != a.size(); ++i) {
  const auto &x = a[i];
  do_something(x);
}
```

□ C.

Let l be of type list<T> for some type T. To traverse l using iterators, we can write

```
for (auto it = l.begin(); it < l.end(); ++it)
  do_something(*it);</pre>
```

 \square D.

Let v be of type vector<T> for some type T. To make a copy of v, we must use a loop like this:

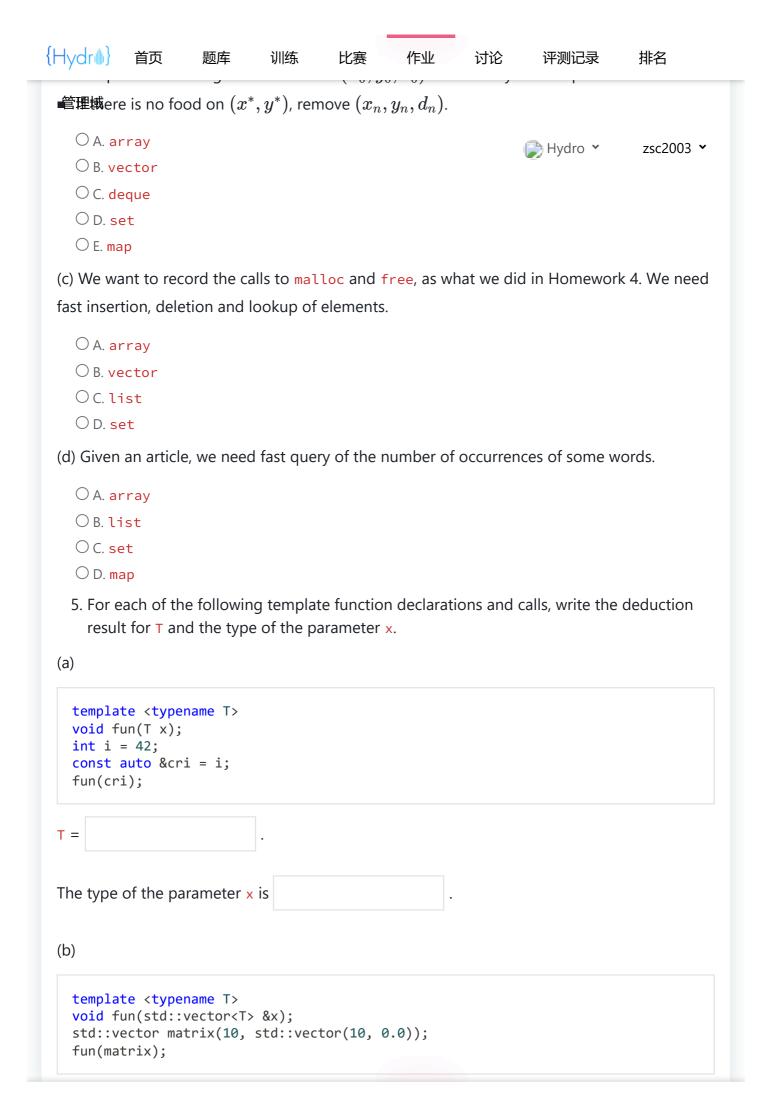
```
vector<T> w;
for (const auto &x : v)
  w.push_back(x);
```

□ E.

Let v be of type vector < T > for some type T. To move-construct a new object w from v, we can use a loop like this;

```
vector<T> w;
for (auto &&x : v)
  w.push_back(std::move(x));
```

- 4. Choose the best containers.
- (a) We want to store a sequence of numbers a_0, a_1, \dots, a_n , where n is determined at runtime. We need fast access of elements given their indexes.
 - O A. array
 - OB. vector
 - OC. list
 - OD. set
- (b) We are writing a *Greedy Snake* game and want to store the body of a snake. The body of a snake can be represented by a sequence of tuples (x_0,y_0,d_0) , \cdots , (x_n,y_n,d_n) , where (x_i,y_i) is the coordinate of the i-th part and $d_i\in\{\uparrow,\downarrow,\leftarrow,\rightarrow\}$ is the moving direction of the i-th part. The snake head is (x_0,y_0,d_0) . Moving the snake one step forward consists of two operations:



```
管理域
The type of the parameter x is
                                                                 😭 Hydro 🗡
                                                                                  zsc2003 ▼
(c)
  template <typename T>
  void fun(T &&x)
  std::string s;
  fun(s);
T =
The type of the parameter x is
(d)
  template <typename T>
  void fun(T &&x);
  std::string s, t;
  fun(s + t);
The type of the parameter \boldsymbol{x} is
(e)
  template <typename T>
  void fun(T x);
  int a[10];
  fun(a);
T =
The type of the parameter x is
(f)
  template <typename T>
  void fun(T &x);
```

管理域	

The type of the parameter \mathbf{x} is

Part 2: Dynarray 4.0

Read the code in dynarray.hpp and answer the following questions. For your convenience, we have marked the question numbers and the choices in the code.

Click this link to download dynarray.hpp. If you encounter any problems, go to Piazza resources to download it.

6. Which of the following is/are correct?

- ☐ A. The assignment operator is both a copy assignment operator and a move assignment operator.
- ☐ B. In the assignment operator, the type of the parameter other is Dynarray<T>.
- C. Note that the cbegin() function calls the begin() function. If a is of type

Dynarray<double>, a.cbegin() will call the non-const version of the begin() function.

- D. Suppose a is an object of type Dynarray < double >. If the non-const version of begin() is not present, a.begin() does not compile.
- 7. Read the following code.

```
#include "dynarray.hpp"

struct X {
    int id;
};

std::ostream & operator << (std::ostream & os, const X & x) {
    return os << "X(" << x.id << ')';
}

int main() {
    Dynarray < X > a(5);
    for (auto i = 0; i != 5; ++i)
        a[i].id = i * i;
    std::cout << a << std::endl;
}</pre>
```

Which of the following is/are correct?

□ A.

Note that the operator< for Dynarray<T> relies on the operator< for T. Since X does not have an overload for operator<, X cannot be the element type of Dynarray and the code above does not

管理域code above compiles and prints [0,1,4,9,16].
□ C. ———————————————————————————————————
The code above compiles and prints [X(0), X(1), X(4), X(9), X(16)].
\Box D.
The code above does not compile if the default constructor of X is deleted, i.e.
<pre>// In class X X() = delete;</pre>
□ E.
The following code does not compile if Y does not have a default constructor.
Dynarray <y> a;</y>
8. Which of the following is/are correct?
□ A.
<pre>Dynarray<t>::iterator is DynarrayIterator<t>.</t></t></pre>
□ B.
<pre>Dynarray<t>::const_iterator is DynarrayIterator<t, true="">.</t,></t></pre>
□ c.
The iterator of Dynarray <t> is a RandomAccessIterator.</t>
\Box D.
Since DynarrayIterator <t, c=""> has a pointer member m_current, it should have a destructor defined as this:</t,>
<pre>// In class DynarrayIterator<t, c=""> ~DynarrayIterator() { delete[] m_current; }</t,></pre>
□ E.
DynarrayIterator is neither copyable nor movable, because it does not have any copy or move operations defined.
9. Note that the <pre>static_assert</pre> at the beginning of <pre>Dynarray</pre> ensures that T must be a non-const type. Which of the following is/are correct?
□ A.
<pre>Dynarray<int>::iterator::value_type is int, and</int></pre>
Dynarray <int>::const_iterator::value_type is const_int, because const_iterator is the iterator with "low-level constness".</int>

```
The subscript operator of DynarrayIterator should have const and non-const overloads like this:

// In class DynarrayIterator<T, C>
T & operator[](difference_type n) {
    return m_current[n];
}
const T & operator[](difference_type n) const {
    return m_current[n];
}
```

10. A template function declared like this

can only be called if the condition some_condition evaluates to true. For example, the following function foo can only be called with an integral argument.

All these metafunctions (std::enable_if, std::is_integral, std::is_same, ...) are defined in <type_traits> and you can find the references here.

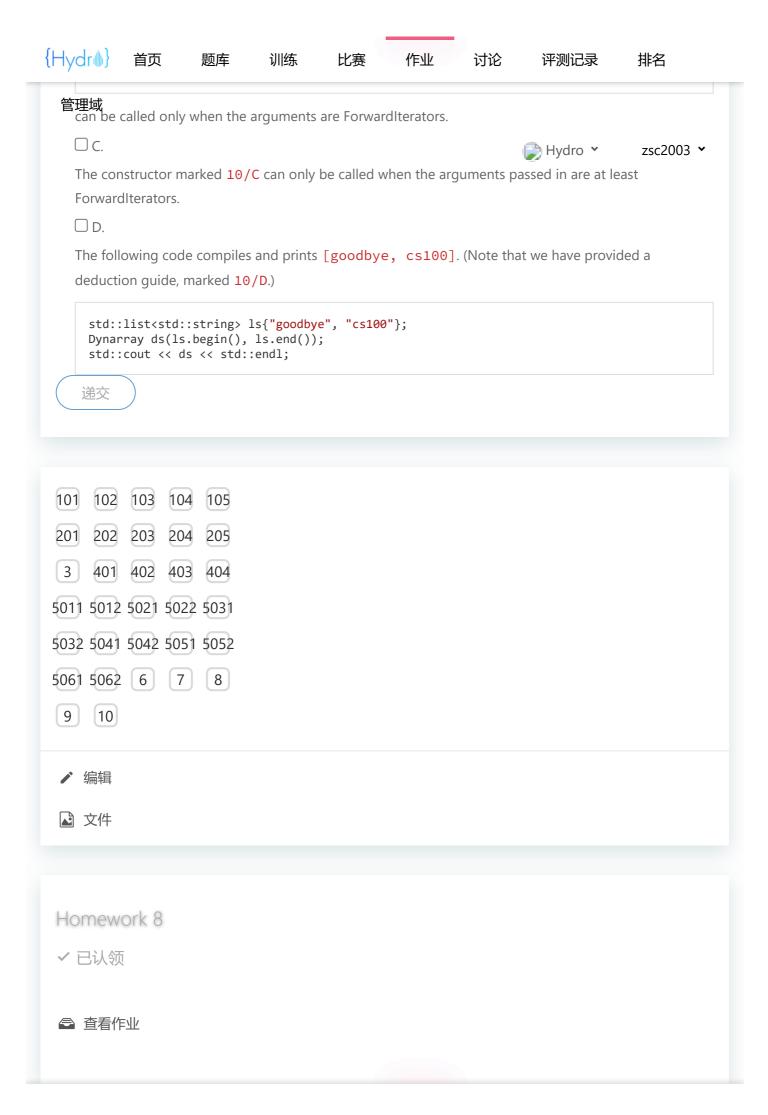
Which of the following is/are correct?

□ A.

The constructor marked 10/A can only be used to construct a const_iterator from an iterator, which is adding the low-level constness.

□ B.

The following constructor





状态

评测队列

服务状态

开发

开源

API

支持

帮助

{Hydrill} 首页 题库 训练 比赛 作业 讨论 评测记录 排名

管理域

关于 联系我们 隐私 服务条款 版权申诉 S Language ^ Legacy mo H Legacy mo Legacy mo Legacy mo Norker 0, 73ms Powered by Hydro v4.9.18 Community