Metaprogramming

Logitech HW01

1. Write a tuple_visiter to print all elements in a tuple.

```
#include <iostream>

struct F

{

   template <typename T>

   void operator()(T&& t) { std::cout << "unexpected type: " << typeid(std::forward<T&&>(t)).name() << std::endl; }

   void operator()(int i) { std::cout << "void F::operator()(int): " << i << std::endl; }

   void operator()(double d) { std::cout << "void F::operator()(double): " << d << std::endl; }

   void operator()(const std::string& s) { std::cout << "void F::operator()(const std::string&): " << s << std::endl; }

};


int main()

{

   auto t = std::make_tuple(10, std::string("Test"), 3.14, "Test");

   F f;

   tuple_visitor(f, t); //please implement this tuple_visitor

   return 0;

}
```

2. Given the following unit test code, complete the implementation of the required structs.

```
#include <iostream>
#include <tuple>
#include <vector>
#include <string>
#include <array>
#include <type_traits>

using namespace std;
//Give implement to the structs below
template <int v> struct Int2Type { enum { value = v }; };
template <class Tuple>                    struct Length;
template <class Tuple, unsigned int index> struct TypeAt;
template <class Tuple, class T>           struct IndexOf;
template <class Tuple, class T>           struct Append;
template <class Tuple, class T>           struct Erase;
template <class Tuple, class T>           struct EraseAll;
template <class Tuple>                    struct NoDuplicates;
template <class Tuple, class T, class U>   struct Replace;
template <class Tuple, class T, class U>   struct ReplaceAll;

int main()
{
   using MyTuple1 = tuple<int, double, string, char, bool, string>;
   using MyTuple2 = tuple<int, double, long, char, bool, long>;
   using MyTuple3 = tuple<int, double, long, bool, bool, long>;
   using MyTuple4 = tuple<int, double, long, bool>;
   using MyTuple5 = tuple<int, double, string, char, bool, string, long long>;
   using MyTuple5 = tuple<int, string, char, bool, string>;
   using MyTuple6 = tuple<int, double, char, bool>;

   static_assert( is_same< ReplaceAll  < MyTuple1, string, long >::type, MyTuple2 >::value );
   static_assert( is_same< Replace     < MyTuple1, char, bool >::type, MyTuple3 >::value );
   static_assert( is_same< NoDuplicates< MyTuple1 >::type, MyTuple4 >::value );
   static_assert( is_same< Length      < MyTuple1 >::value, Int2Type<3>::value >::value );
   static_assert( is_same< TypeAt      < MyTuple1, 2>::type, double >::value );
   static_assert( is_same< IndexOf     < MyTuple1, string>::value, Int2Type<2>::value >::value );
   static_assert( is_same< Append      < MyTuple1, long long >::type, MyTuple5 >::value );
   static_assert( is_same< Erase       < MyTuple1, double>::type, MyTuple5 >::value );
   static_assert( is_same< EraseAll    < MyTuple1, string>::type, MyTuple6 >::value );
   return 0;
}
```