

Template

Problem 1: Will the code below compile?

```
#include <iostream>
using namespace std;

template<typename T>
T minimum(const T& a, const T& b)
{
    return (a < b) ? a : b;
}

int main()
{
    int x = minimum(3, 5);
    cout << "x is: " << x << endl;
    double y = minimum(3.2, 5.2);
    cout << "y is: " << y << endl;
    int z = minimum(3.2, 5.2);
    cout << "z is: " << z << endl;
    string w = minimum(3.2, 5.2);
    cout << "w is: " << w << endl;
    int m = minimum(3, 4.7);
    cout << "m is: " << m << endl;
}
```

Problem 2: What is the output of the code?

```
#include <iostream>
using namespace std;

template<typename T>
T minimum(const T& a, const T& b)
{
    cout << "Use template minimum()" << endl;
    return (a < b) ? a : b;
}

double minimum(const double& a, const double& b){
    cout << "Use double minimum()" << endl;
    return (a < b) ? a : b;
}

int main()
{
    int x = minimum(3, 5);
    cout << "x is: " << x << endl;
    double y = minimum(3.2, 5.2);
    cout << "y is: " << y << endl;
    return 0;
}
```

Problem 3: Will the code below compile?

```
#include <iostream>
using namespace std;

class Animal
{
public:
    Animal(int weight):m_weight(weight){};
    int getWeight() const {return m_weight;};
private:
    int m_weight;
};

template<typename T>
bool lessThan(const T& a, const T& b)
{
    return (a < b);
}

int main()
{
    Animal a(10);
    Animal b(15);
    cout << lessThan(a, b) <<endl;
    return 0;
}
```

STL(Standard Template Library)**How to print all elements of vector<int> or list<int>?**

```
list<int> li;
for ( _____ )
{
    cout<<_____<<endl;
}
```

What's the problem with the code below?

```
for (vector<int>::iterator it = li.begin(); it != li.end(); it++)
{
    if(*it > 2)
        li.erase(it);
}
```

Inheritance

What's the output?

```
#include <iostream>
using namespace std;
class A
{
public:
    A(){cout << "A()" << endl;}
    A(int x){cout<< "A(" << x << ")" << endl;}
    ~A(){cout << "~A()" << endl;}
};

class B
{
public:
    B(){cout << "B()" << endl;}
    B(int x):m_a(x){cout << "B(" << x << ")" << endl;}
    ~B() {cout << "~B()" << endl;}
private:
    A m_a;
};

class C:public A
{
public:
    C():A(10), m_b2(5){ cout << "C()" << endl;}
    ~C(){ cout << "~C()"<< endl;}
private:
    B m_b1;
    B m_b2;
};

int main() {
    C c;
}
```