```
 1 #include <iostream>
 2 #include <vector>
 3
 4 template<typename T> void p(T x) { std::cout << x; }
 5
 6 int main() {
 7     int my_data[] = {3,7,8,2,5};
 8     std::vector<int> values(&my_data[0], &my_data[sizeof(my_data)/sizeof(int)]);
 9     for(int i=0; i<values.size(); ++i)
10         p(values[i]);
11 }
```

What will this code print out? Please critizise the code. What will happen if we use std::list instead of std::vector? Please provide an alternative implementation of line 9-10.

```
 1 #include <iostream>
 2 #include <algorithm>
 3 #include <vector>
 4 #include <string>
 5
 6 struct print {
 7     std::string prefix_;
 8     print( std::string prefix ) : prefix_(prefix) {}
 9     void operator() (int i) {
10         std::cout << prefix_ << i << std::endl;
11     }
12 };
13
14 int main() {
15     std::vector<int> v;
16     v.push_back(1);
17     v.push_back(2);
18     v.push_back(3);
19     std::for_each( v.begin(), v.end(), print("i=") );
20 }
```

what might happen if you try to compile, link and run this program?

```
 1 #include <iostream>
 2 #include <list>
 3 #include <algorithm>
 4
 5 template<typename T> void p(T x) { std::cout << x; }
 6
 7 template<class C> typename C::value_type sum(const C & c) {
 8     typename C::value_type s = 0;
 9     for (typename C::const_iterator i = c.begin(); i != c.end(); ++i )
10         s += *i;
11     return s;
12 }
13
14 int main() {
15     typedef long data_type;
16     data_type my_data[] = {3,7,8};
17     typedef std::list<data_type> container;
18     container values(&my_data[0], &my_data[sizeof(my_data)/sizeof(my_data[0])]);
19     for_each(values.begin(), values.end(), p<data_type>);
20     p(sum<container>(values));
21 }
```

What might happen if you try to compile, link and run this program? How many lines do we have to change if we want to work with a vector of ints instead? How can we make this code print "87318" instead? Why is the keyword typename on line 8 and 9 needed?

```
 1 #include <iostream>
 2 #include <queue>
 3 #include <string>
 4
 5 template<typename T> void p(T x) { std::cout << x; }
 6
 7 class Message {
 8     std::string msg_;
 9     int priority_;
10 public:
11     Message(std::string msg, int priority) : msg_(msg), priority_(priority) {}
12     bool operator<(const Message & m) const { return priority_ < m.priority_; }
13     std::string msg() const { return msg_; }
14     int priority() const { return priority_; }
15 };
16
17 std::ostream & operator<<(std::ostream & ostm, const Message & m) {
18     return ostm << m.msg() << '(' << m.priority() << ')';
19 }
20
21 int main() {
22     std::priority_queue<Message> q;
23     q.push(Message("Foo",4));
24     q.push(Message("Bar",2));
25     q.push(Message("Gaz",3));
26     q.push(Message("Daz",5));
27     q.push(Message("Boo",8));
28     while ( !q.empty() ) {
29         p(q.top());
30         q.pop();
31     };
32 }
```

what might happen if you try to compile, link and run this program?

```
 1 #include <iostream>
 2 #include <string>
 3 #include <map>
 4
 5 int main() {
 6     std::map<std::string,int> m;
 7     m["foo"] = 2;
 8     m["foo"] = 4;
 9     m["bar"] = 8;
10     m["gaz"] = 5;
11     m["gaz"]++;
12     for( std::map<std::string,int>::iterator i = m.begin();
13          i != m.end(); ++i) {
14         std::cout << (*i).first << " " << (*i).second << std::endl;
15     }
16
17     std::pair<std::string,int> b("bar",1);
18     std::pair<std::map<std::string,int>::iterator,bool> p = m.insert(b);
19     std::cout << std::boolalpha << p.second << std::endl;
20     std::cout << m["bar"] << std::endl;
21 }
```

what might happen if you try to compile, link and run this program?

```
 1 #include <iostream>
 2 #include <bitset>
 3 #include <string>
 4
 5 int main() {
 6     std::bitset<8> b = 0x01;
 7
 8     b << 1;
 9     std::cout << b << std::endl;
10
11     b |= std::bitset<8>(std::string("11110000"));
12     std::cout << b << std::endl;
13
14     std::cout << b.count() << std::endl;
15 }
```

what might happen if you try to compile, link and run this program?

```
 1 #include <iostream>
 2 #include <algorithm>
 3 #include <vector>
 4 #include <list>
 5 #include <iterator>
 6
 7 int main() {
 8     std::vector<int> v1;
 9     v1.push_back(1);
10     v1.push_back(4);
11     v1.push_back(3);
12     v1.push_back(4);
13     v1.push_back(9);
14     std::list<int> v2;
15     std::copy( v1.begin(), v1.end(), front_inserter(v2));
16     std::copy( v2.rbegin(), v2.rend(), std::ostream_iterator<int>(std::cout) );
17 }
```

what might happen if you try to compile, link and run this program?

```
 1 #include <iostream>
 2 #include <algorithm>
 3
 4 template<typename T> void p(T x) { std::cout << x; }
 5
 6 class my_array {
 7 public:
 8     typedef short value_type;
 9     typedef value_type * iterator;
10     typedef value_type & reference;
11     reference operator [] (ptrdiff_t i) { return v[i]; }
12     iterator begin() { return v; }
13     iterator end() { return v+size(); }
14     size_t size() const { return sizeof(v) / sizeof(value_type); }
15     my_array() : v() {}
16 private:
17     value_type v[4];
18 };
19
20 int main() {
21     my_array m;
22     m[2] = 4;
23     m[3] = 2;
24     std::for_each( m.begin() + 2, m.end(), p<my_array::value_type> );
25 }
```

what might happen if you try to compile, link and run this program?

```
 1 #include <iostream>
 2 #include <algorithm>
 3 #include <vector>
 4 #include <functional>
 5
 6 class Foo {
 7     int v;
 8 public:
 9     Foo(int i) : v(i) {}
10     void print() { std::cout << v; }
11 };
12
13 int main() {
14     std::vector<Foo> v;
15     v.push_back(Foo(1));
16     v.push_back(Foo(2));
17     v.push_back(Foo(3));
18     std::for_each( v.begin(), v.end(), Foo::print );
19 }
```

This code does not compile. How to fix it so that it prints 123?

```
 1 #include <iostream>
 2 #include <algorithm>
 3 #include <vector>
 4
 5 bool equal_to_4_func(int i) {
 6     return i == 4;
 7 }
 8
 9 struct equal_to_4_class {
10     bool operator()(int i) { return i == 4; }
11 };
12
13 int main() {
14     std::vector<int> v;
15     v.push_back(1);
16     v.push_back(4);
17     v.push_back(3);
18     v.push_back(4);
19     v.push_back(9);
20     std::cout << std::count_if( v.begin(), v.end(), equal_to_4_func );
21     std::cout << std::count_if( v.begin(), v.end(), equal_to_4_class() );
22 }
```

what might happen if you try to compile, link and run this program?

```
 1 #include <iostream>
 2 #include <algorithm>
 3 #include <vector>
 4 #include <functional>
 5
 6 template <int v> bool equal_to_func_templ(int i) {
 7     return i == v;
 8 }
 9
10 template <int v> struct equal_to_class_templ {
11     bool operator()(int i) const { return i == v; }
12 };
13
14 struct my_equal_to : public std::binary_function<int, int, bool> {
15     bool operator() (int a, int b) const { return a == b; }
16 };
17
18 int main() {
19     std::vector<int> v;
20     v.push_back(1);
21     v.push_back(4);
22     v.push_back(3);
23     v.push_back(4);
24     v.push_back(9);
25     using namespace std;
26     cout << count_if( v.begin(), v.end(), equal_to_func_templ<4> ) << endl;
27     cout << count_if( v.begin(), v.end(), equal_to_class_templ<4>() ) << endl;
28     cout << count_if( v.begin(), v.end(), bind2nd(equal_to<int>(),4) ) << endl;
29     cout << count_if( v.begin(), v.end(), bind1st(my_equal_to(),4) ) << endl;
30 }
```

what might happen if you try to compile, link and run this program?