STL QUIZ

Attempt the questions before you look at the answers.
This quiz is not graded.

1. Can you find the bug in this code: (assume string and cout are the ones in std namespace)

```
int main( int argc, char ** argv )
{
   char  * ptr1 = NULL;
   bool  conditionA = true;

   if (conditionA)
   {
      std::string  s1 (“conditionA is true”);

      ptr1 = s1.c_str();

      // use s1, etc.
   }
   else
   {
      ptr1 = “conditionA was not true”;
   }

   cout   <<  ptr1 << endl;

   return 0;
}
```

1 Ans

Can you find the bug in this code: (assume string and cout are the ones in std namespace)

```
int main( int argc, char ** argv )
{
   char  * ptr1 = NULL;
   bool  conditionA = true;

   if (conditionA)
   {
      std::string  s1 ("conditionA is true");

      ptr1 = s1.c_str();

      // use s1, etc.
   }
   else
   {
      ptr1 = "conditionA was not true";
   }

   cout   <<  ptr1 << endl;

   return 0;
}
```

The address pointed to by ptr1 in the if / then block is not valid outside the if/then block since the string s1 will get destructed when control comes out of the if/then block.

2. Assume MyString is some class that you have defined.

```
#include <vector>

std::vector< MyString > vi (someSize);
```

This statement will not compile if class MyString … ?

2 Ans

Assume MyString is some class that you have defined.

```
#include <vector>

std::vector< MyString > vi (someSize);
```

This statement will not compile if class MyString … ?

Won’t compile if class MyString does not define a default constructor.
If you do not want the default constructor to be called for every element of this vector when you define it, you can use the `reserve` method on the vector to allocate space, and in that case, only memory gets allocated inside the vector, and the constructor is not called.

3. What improvement in performance can you do below? Assume MyClass is some class that you have defined.

```
int size = someCollection.size();

std::vector < MyClass>   vec1;

for ( int ii = 0; ii < size; ++ ii )
{
      vec1.push_back( someCollection[ ii ] );
}
```

3 Ans

What improvement in performance can you do below? Assume MyClass is some class that you have defined.

```
int size = someCollection.size();

std::vector < MyClass>   vec1;

for ( int ii = 0; ii < size; ++ ii )
{
   vec1.push_back( someCollection[ ii ] );
}
```

Since we know the number of elements we want to push onto the vector, we should call `reserve` to allocate enough space such that the vector will not need to do any reallocations (and subsequent copying) during the loop. This would look like:

```
int size = someCollection.size();

std::vector < MyClass>   vec1;
vec1.reserve( size );
for ( int ii = 0; ii < size; ++ ii )
{
   vec1.push_back( someCollection[ ii ] );
}
```

// Note that if you define as: `std::vector < MyClass>   vec1(size);`
// Then the default constructor of MyClass will get called for each of the size elements in the vector,
// whereas in the case of the reserve call, only enough memory is allocated. If the default
// constructor is not defined for MyClass, this statement will give a compile error.

4. What improvement in performance can you do below?

```
for ( someIterator = someCollection.begin();
      someIterator != someCollection.end();
      someIterator ++ )
{
      // … some code here
}
```

4 Ans

What improvement in performance can you do below?

```
for ( someIterator = someCollection.begin();
      someIterator != someCollection.end();
      someIterator ++ )
{
   // … some code here
}
```

Should use pre increment of the iterator, since as we read in the lesson, that is a little faster than the post increment. So it should be:

```
for ( someIterator = someCollection.begin();
      someIterator != someCollection.end();
      ++ someIterator )
{
   // … some code here
}
```