C++FA 6.1 PRACTICE FINAL EXAM

This practice final exam covers sections C++FA 2.1 through C++FA 4.1 of ***C++ with Financial Applications*** by Ben Van Vliet, available at **www.benvanvliet.net**.

1.) A T is a/an:

a.) Value type.
b.) Reference type.
c.) Whatever type.
d.) Undefined type.
e.) None of the above.

2.) A library that is linked at compile time is a:

a.) Dynamically linked library.
b.) Static library.
c.) Class library.
d.) Function library.
e.) None of the above.

3.) To deal with the problem of creating objects without specifying the exact class of object that will be created, we use:

a.) An observer.
b.) A factory.
c.) A friend.
d.) A template.
e.) None of the above.

4.) Algorithms in the STL sense mean:

a.) Functions.
b.) Template classes.
c.) Collections and iterators.
d.) Sequential containers.
e.) None of the above.

5.) A Singleton should have a private copy constructor so that:

a.) Instances are not implicitly created.
b.) No one can explicitly create more than one instance.
c.) Initially, no instance exists.
d.) If a copy is made, it will have the same values.
e.) None of the above.

6.) **A pair is:**

a.) A key and a collection.
b.) A combination of reference types.
c.) Two closely related instances.
d.) A unique identifier and an object.
e.) None of the above.

7.) **A smart pointer:**

a.) Contains the best of value and reference types.
b.) Overrides operators defined in the base class.
c.) Deletes the pointer but not the object.
d.) Enables automated creation and destruction.
e.) None of the above.

8.) **Consider the following line of code:**

```
vector< g * > *m = new vector< g * >;
```

What's going on here?

a.) Creating a value type vector m that contains reference types of g.
b.) Creating a reference type vector m that contains reference types of g.
c.) Creating a value type vector m that contains value types of g..
d.) Creating a reference type vector m that contains value types of g..
e.) None of the above.

9.) **Consider the following lines of code:**

```
class MyClass
{
private:
      MyClass() {}
};
```

What's going on here?

a.) Class definition with constructor prototye.
b.) Definition of an uninstantiatable class.
c.) .cpp file code for MyClass.
d.) Abstract class MyClass declaration.
e.) None of the above.

10.) Consider the following line of code:

```
typedef x< int > X;
```

What's going on here?

a.) X is an x of int.
b.) X is an int.
c.) Compile error.
d.) X is an instance of the template class x.
e.) None of the above.

11.) Consider the following line of code:

```
template< class T >
Node< T > *MyList::foo( const T & ) {…}
```

What's going on here?

a.) MyList method that accepts a T by reference.
b.) Foo is a function pointer to a function in class Node< T >.
c.) Template class MyList prototype of static method foo.
d.) Node< T > class method returns a pointer to a T.
e.) None of the above.

12.) What happens when the program compiles?

a.) Linker creates machine language code, then compiles executable.
b.) Object files are linked to create .exe file.
c.) Compiler cleans all corrupt files then rebuilds.
d.) Only out of date files are not recompiled first, then linked.
e.) None of the above.

13.) Given the following function:

```
template < class T >
T f( int T )
{
      return ++T;
}
```

What is the output from the following code?

```
int a = 2;
int b = f( a );
cout << a << " " << b << endl;
```

a.) 2 2
b.) 2 int
c.) 2 T
d.) 2 3
e.) None of the above.

For Questions 14-17, assume the following class definitions:

```
template < class T >
class MyClass
{
private:
      T myValue;

public:
      MyClass( T );
      T get_Value();
};

template < class T >
MyClass< T >::MyClass( T x ) : myValue ( x )
{
}

template < class T >
T MyClass< T >::get_Value()
{
      return myValue;
}

////////////////////////////////////////////////////////////////////

template< class T >
class smart_ptr
{
private:
      T *ptr;

public:
      smart_ptr( T *p = 0 ) : ptr( p )
      {
      }
      ~smart_ptr()
      {
            delete ptr;
      }
      T &operator*()
      {
            return *ptr;
      }
      T *operator->()
      {
            return ptr;
      }
};

typedef smart_ptr< int > INT;
```


14.) Which line of code would correctly instantiate a MyClass containing an INT?

a.) `MyClass< INT * > m( new INT( new int( 3 ) ) );`
b.) `MyClass< INT > m( new INT( new int( 3 ) ) );`
c.) `MyClass< INT * > m( INT( new int( 3 ) ) );`
d.) `MyClass< INT * > m( new INT( 3 ) );`
e.) None of the above.

15.) Given a value type instance of MyClass with a myValue of 3 and a reference type instance of INT, which line of code would print out 3?

a.) `cout << *( m.get_Value() ) << endl;`
b.) `cout << **( m->get_Value() ) << endl;`
c.) `cout << **( m.get_Value() ) << endl;`
d.) `cout << *( m->get_Value() ) << endl;`
e.) None of the above.

16.) Which line of code would correctly instantiate a smart_ptr to an instance of MyClass?

a.) `smart_ptr< MyClass< int > * > n( new MyClass< int >( 4 ) );`
b.) `smart_ptr< MyClass< int * > > n( new MyClass< int >( 4 ) );`
c.) `smart_ptr< MyClass< int > > n( MyClass< int >( 4 ) );`
d.) `smart_ptr< MyClass< int > > n( new MyClass< int >( 4 ) );`
e.) None of the above.

17.) Given a value type instance of smart_ptr containing a value type instance of MyClass with a myValue of 4, which line of code prints out 4?

a.) `cout << *( n.get_Value() ) << endl;`
b.) `cout << n->get_Value() << endl;`
c.) `cout << n.get_Value() << endl;.`
d.) `cout << **n->get_Value() << endl;`
e.) None of the above.

18.) What is the output from the following code?

```
vector< int >  a;
for ( int x = 0; x < 5; x++ )
{
      a.push_back( x );
      a.push_back( x + 1 );
}
a.pop_back();

cout << a[ 8 ] << endl;
```

a.) 8
b.) 3
c.) 5
d.) 4
e.) None of the above.

19.) Given the following code, what is in v2[0]? (Recall that % is the modulo operator. It finds the remainder of one number divided by another.)

```
vector< int > v1, v2;
for( int x = 0; x < 5; x++ )
{
      v1.push_back( x );
}

vector< int >::reverse_iterator m_Reverse;

for( m_Reverse = v1.rbegin(); m_Reverse != v1.rend(); m_Reverse++ )
{
      if( *m_Reverse % 2 == 0 )
      {
            v2.push_back( *m_Reverse );
      }
}
cout << v2[ 0 ] << endl;
```

a.) 5
b.) 4
c.) 3
d.) 2
e.) None of the above.

20.) Given the following function:

```
void print( int a )
{
      cout << a << endl;
}
```

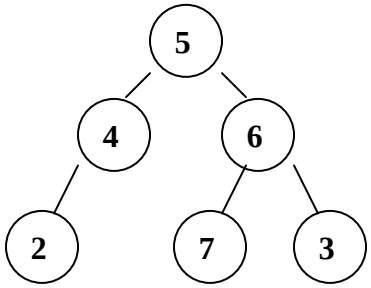
What is the output of the following code?

```
list< int > m_List;

m_List.push_front( 2 );
m_List.push_front( 4 );
m_List.push_front( 3 );

m_List.sort();
m_List.insert( m_List.begin(), 6 );

for_each( m_List.begin(), m_List.end(), print );
```

a.) 2 3 4 6
b.) 6 2 4 3
c.) 6 2 3 4
d.) 2 4 3 6
e.) None of the above.

21.) Given the following tree structure with nodes:

LEFT **RIGHT**




```
struct Node
{
      Node( int v ) : Value( v )
      {
            Right = NULL;
            Left = NULL;
      }
      int Value;
      Node *Right;
      Node *Left;
};
```

What is the return value of the following function, where root points to the tree?

```
int NodeSum( Node *root )
{
   if( root == NULL )
   {
      return 0;
   }
   else
   {
      return root->Value + NodeSum( root->Right ) - NodeSum( root->Left );
   }
}
```

a.) 0
b.) 7
c.) 27
d.) 5
e.) None of the above.

22.) If MyList is a sequential collection of Nodes, what does the following member function do?

```
void MyList::Function_A()
{
      if ( begin == 0 )
      {
            return;
      }
      else
      {
            Node *y = begin;

            if( begin->get_NextPtr() == 0 )
            {
                  begin = 0;
            }

            while ( y->get_NextPtr() != 0 )
            {
                  y = y->get_NextPtr();
            }
            delete y;

            return;
      }
}
```

a.) Removes a Node from the back of the list.
b.) Removes a Node from the front of the list.
c.) Pops a Node from the middle of the list.
d.) Pops a Node from the front of the list.
e.) None of the above.

23.) What is the acceptable ways to create an instance of the list class?

```
1.)   list< int > *m_List = new list;
2.)   list< int * > *m_List = new list< int * >;
3.)   list< int > m_List;
4.)   list m_List;
```

a.) 1. and 2.
b.) 1. and 3.
c.) 2, 3 and 4.
d.) 2 and 3.
e.) None of the above.

24.) In C++, what is meant by a key?

a.) A table of T's which determine memory locations.
b.) A object hashed to produce a memory location.
c.) The second element of a C++ template type pair.
d.) A typedef for adding elements to a hash_map.
e.) None of the above.

25.) With respect to the singleton design pattern, which is false?

a.) The intent of a singleton is to ensure that a class has only one instance.
b.) The singleton allows for strict control over how and when clients access it.
c.) The sole instance of a singleton should be extensible by subclassing.
d.) The class itself is responsible for keeping track of its sole instance.
e.) None of the above.

26.) In OOP how would you describe encapsulation?

a.) The conversion of one type of object to another.
b.) The runtime resolution of method calls.
c.) The declaration of data.
d.) The separation of interface and implementation.
e.) None of the above.

27.) Which of the following is not a way to increase the execution speed of C++ code?

a.) Initialize variables upon declaration.
b.) Put most frequent cases first in a switch-case statement.
c.) Use inline for small functions.
d.) Use nameless objects.
e.) None of the above.

28.) Given the following code, what are the pairs in the hash_map?

```
hash_map< int, int > m_Map;
typedef pair<int, int> Int_Pair;

m_Map.insert( Int_Pair( 1, 100 ) );
m_Map.insert( Int_Pair( 2, 100 ) );
m_Map.insert( Int_Pair( 3, 200 ) );
m_Map.insert( Int_Pair( 2, 200 ) );
m_Map.insert( Int_Pair( 3, 300 ) );
m_Map.insert( Int_Pair( 8, 100 ) );
```

a.) 1 100; 8 100; 2 100; 3 200.
b.) 1 100; 2 100; 3 200; 2 200; 3 300; 8 100.
c.) 1 100; 3 200; 3 300.
d.) 1 100; 2 200; 3 300; 8 100.
e.) None of the above.

For Questions 29-31, assume the following class definitions:

```
template < class T >
class MyClass
{
private:
      T &myValue;

public:
      MyClass( T & );
      ~MyClass(void);

      T get_Value();
};

template < class T >
MyClass< T >::MyClass( T &x ) : myValue ( x )
{
}
template < class T >
MyClass< T >::~MyClass(void)
{
}
template < class T >
T MyClass< T >::get_Value()
{
      return myValue;
}

//////////////////////////////////////////////////////////////////////////

class Element
{
private:
      int myData;
public:
      Element( int );
      ~Element(void);
      int get_Data();
};

Element::Element( int x )
{
      myData = x;
}
Element::~Element(void)
{
      cout << myData << " is dying!" << endl;
}
int Element::get_Data()
{
      return myData;
}
```

29.) What is the output from the following code?

```
int x = 4;
int y = 5;
MyClass< int > m_Obj1( y );
MyClass< int > *m_Obj2 = new MyClass< int >( x );
cout << m_Obj1.get_Value() << " ";
cout << m_Obj2->get_Value() << endl;
```

a.) 4 5
b.) 5 4
c.) Compile error: 'initializing' : cannot convert from 'MyClass<T> *' to 'MyClass<T>'.
d.) Compile error: left of '.get_Value' must have class/struct/union.
e.) None of the above.

30.) Not including the constructor and destructor, given the following line of code, which line will correctly print out 4?

```
MyClass< Element > m_Obj( Element( 4 ) );
```

a.) `cout << m_Obj.get_Data() << endl;`
b.) `cout << m_Obj.get_Value()->get_Data() << endl;`
c.) `cout << m_Obj.get_Value().get_Data() << endl;`
d.) `cout << m_Obj.get_Value() << endl;`
e.) None of the above.

31.) Which line of code will successfully create a hash_map where the first element is an integer and the second is an instance of MyClass?

a.) `hash_map< int, MyClass > m_Map;`
b.) `hash_map< int, MyClass< Element * > * > m_Map;`
c.) `hash_map< int, MyClass * > m_Map;`
d.) `hash_map< int, MyClass< Element > > *m_Map;`
e.) None of the above.

32.) What is the output from the following code?

```
list <int> m_List;
list <int>::iterator m_Result;

for ( int i = 10; i <= 100; i += 10 )
{
      m_List.push_back( i );
}

m_Result = find( m_List.begin(), m_List.end(), 60 );

if  ( m_Result == m_List.end( ) )
{
      cout << "Not Found." << endl;
}
else
{
      cout << "Found: " << *m_Result << endl;
}
```

a.) Not Found.
b.) Found: 60
c.) Found: 6
d.) Found: true
e.) None of the above.

33.) What is the output from the following code?

```
typedef pair< int, double > MyPairType;
MyPairType m_Pair( 1, 2.71828 );
cout << m_Pair.first << "  " << m_Pair.second << endl;
```

a.) 1 2.71828
b.) true true
c.) int double
d.) first second
e.) None of the above.

34.) What is the base memory structure (base_mat) of the matrix class?

a.) Vector.
b.) Array of arrays.
c.) Hash_map
d.) List.
e.) None of the above.

35.) What is the purpose of the Mersenne Twister algorithm?

a.) Generate standard normal random numbers.
b.) Generate 0 to 1 random numbers.
c.) Generate Monte Carlo simulation outcomes.
d.) Generate matrix multiplications.
e.) None of the above.

36.) What is the purpose of the following code?

```
Option::Option( char *symbol, char *putcall ) : Instrument( symbol )
```

a.) Declares the Option class as derived from Instrument.
b.) Prototypes the constructor for the Option class.
c.) Starts the constructor definition by calling the base class constructor.
d.) Defines the constructor in-line.
e.) None of the above.

37.) Given the following function:

```
double new_present_value( const vector< double > *times,
                          const vector< double > *cashflows,
                          const double &r )
{
      double p = 0;
      for ( int i = 0; i < times->size(); i++ )
      {
            .....
      }
      return p;
};
```

The ….. should be replaced by which line of code?

a.) p += (*cashflows)[i] / pow((1 + r), (*times)[i]);
b.) p += cashflows[i] / pow((1 + r), times[i]);
c.) p += *cashflows)[i] / pow((1 + r), *times[i]);
d.) p += cashflows->get(i) / pow((1 + r), times->get(i));
e.) None of the above.

38.) The fastest way to pass parameters to functions is:

a.) By Value.
b.) By Reference.
c.) By Pointer.
d.) By Template.
e.) None of the above.

39.) Which of the following is fastest?

a.) `x = x + 1;`
b.) `x++;`
c.) `x += 1;`
d.) `++x;`
e.) None of the above.

40.) Which of the following for loops will execute fastest?

a.) `for( int x = 0; x < 100; x++ )`
b.) `for( double x = 0; x < 100; x++ )`
c.) `for( int x = 100; x > 0; x++ )`
d.) `for( double x = 100; x > 0; x++ )`
e.) None of the above.

41.) The purpose of this analysis tool is to determine which sections of a program to optimize in order to increase its overall speed, decrease its memory requirement or sometimes both?

a.) Analyzer.
b.) Profiler.
c.) Linker.
d.) Refactorer.
e.) None of the above.

42.) Optimizing the use of objects as return values likely should include the use of which?

a.) Smart pointers.
b.) Reference types.
c.) Template class.
d.) Copy constructor and destructor.
e.) None of the above.

43.) The observer pattern dereferences a function pointer when…

a.) A state change occurs in the subject.
b.) An instance is created by the subject.
c.) A state change occurs in the observer.
d.) An event flag is raised.
e.) None of the above.

44.) Why should we always declare a base class destructor as virtual?

a.) So that overriding will always call the base class destructor.
b.) So that only the derived class definition runs.
c.) So that both definitions will run in order.
d.) So that only the base class definition runs.
e.) None of the above.

45.) What should be done to initialize a variable to its largest value?

a.) Define it as 9999999.
b.) Include the limits library.
c.) Set it equal to `inf()`.
d.) Call a math.h function.
e.) None of the above.

46.) Regarding the matrix class, which is false:

a.) Its use of return by value and implicit object creation means it runs slow.
b.) It makes use of overloaded operators.
c.) It can perform matrix inversion.
d.) Its part of the QuantLib library.
e.) None of the above.

47.) With every use of a memory allocation function (malloc), what function should be used to release allocated memory which is no longer needed?

a.) New.
b.) Free.
c.) Delete.
d.) Remove.
e.) None of the above.

48.) Which STL class is essentially an wrapper around a character array?

a.) vector.
b.) hash_map.
c.) string.
d.) list.
e.) None of the above.

49.) What is the output of the following line of code?

```
printf( "atoi( \"%s\" ) = %i\n", "32", 32 );
```

a.) atoi(32) = 32
b.) atoi(\\ ) = 32
c.) atoi(s) = 32
d.) atio(s) = i 32 32
e.) None of the above.

50.) The scope resolution operator (::), is used to do all of the following except which?

a.) Define a member function outside of the class prototype.
b.) Call a static method.
c.) Define a static member.
d.) Call an instance method.
e.) None of the above.

