# FORM 2 (put name, form, and section number on test!!!)

## EECS 162 Exam II

### True (A) / False (B) (3 pts)

1. A circumstance in C++ where an exception is thrown must be an error. F
2. In the implementation of a `class` template function, in a separate file, the `class` name before the scope resolution operator is just the `class` name, with no additional syntax. F Roomba<T>::
3. An uncaught exception in C++ is ignored. F
4. In C++ an exception object must be derived from the exception class defined in the standard library. F (can be any type)

   Don't like
5. An exception is signaled or raised with the keyword ~~`catch`~~ throw followed by an exception object that may be of any C++ type. F
6. `Virtual` functions are implemented with a table look up that is done at run time. T
7. A program can continue to run after an exception has been thrown and caught. T
8. A class that has a pure virtual member function is called a ~~concrete~~ abstract base class. F
9. This is legal code. F

```
class B
{
public:
  // . . .
  virtual void f() = 0;
};
int main() { B b1, b2; /*. . .*/ }
```

← pure virtual

10. A pointer to objects of a derived class can be assigned pointers to objects of the base class in the inheritance hierarchy. F child *c = new Parent ?
11. When an exception is thrown, the ~~function finishes its execution~~, its value is returned, then control is transferred to the catch block. F
12. It is legal to have all member functions of a class be pure virtual functions. T
13. To instantiate and call, a template function requires special syntax.

14. In the template prefix, template<~~class~~ typename T> the keyword class means that the type parameter T must be of class type. F

15. It is possible to have more than one type parameter in a template definition. T T1, T2

16. Suppose the swapValues template is instantiated as follows:

```
int x = 2, y =3;
swapValues(x, y);
// use x and y
x = 4;  y =5;
swapValues(x, y);
// use x and y
```

F

The compiler generates code for ~~two~~ one copies of the swapValues template.

17. To declare an iterator, one must #include the proper header file, then specify the container type and use that with the scope resolution operator, ::, to qualify the inner type iterator, to declare the iterator variable, as in T

```
#include <vector>
std::vector<int>::iterator myIterator;.
```

18. ~~An input stream is a stream of data flowing from your program, either to a file, or to the keyboard.~~

19. ~~cout has type ostream, i.e., is an output object.~~

20. ~~When you write~~

```
ifstream inStream;
inStream.open(''infile.dat'');
```

~~the file, infile.dat must be located in the directory where the program is being run.~~

## Multiple Choice (3 pts)

21. Which of the following can be virtual?

a) Constructors

[✓ b)] Ordinary functions

c) Friend functions

d) Static functions

22. A C++ exception object may be an int, a char or a double, but unlimited information to be sent when an exception is thrown may be put in

a) The catch block

b) The try block

c) The throw block

[✓ d) An exception object]

e) The function that throws the exception

23. Which of the following operations is not allowed by bidirectional iterators? constant iterator

a) Prefix operator* to make available the container element for use as l-value or r-value.

[✓ b) Overloaded operator+ to add an int value to the iterator to move the place the iterator points forward by the argument number of elements.] random iterator

c) Overloaded operator++ to move the place the iterator points forward by one element. forward/ + bidir

d) Overloaded operator-- to move the place the iterator points backward by one element. bidirection

e) Overloaded operator== and operator!= to determine whether two iterators point to the same element.

24. The expression, $4N^2+2N+1$ is

a) Linear (degree 1)

[✓ b) Quadratic (degree 2)]

c) Cubic (degree 3)

d) Logarithmic ( O(log N) )

25. In the template prefix, template<class T>, what kinds of variables is the parameter T?

a) T must be a type built into C++ such as int or double.

b) T must *not* be a class.

c) T *must* be a class.

[✓ d) T can be any type, whether built into C++ or programmer defined.]

e) T can be any type, whether built into C++ or programmer defined, but subject to restrictions.

26. I have an algorithm that runs in $O(N^2)$, where n is the size of the problem. For N = 100, the time the algorithm runs is 1 minute. How long does the algorithm take for N=1000?

$$\frac{100^2}{1000^2} = \frac{1\ \text{min}}{X}$$

$$\frac{1000000\ 1\ \text{min}}{10000} \qquad \frac{100\,00X}{10000}$$

a) Same time

b) 10 minutes

[✓ c)] 100 minutes

d) 1000 minutes

e) You haven’t given enough information. I can’t tell.

27. I have a pointer, `nodePtr` to a node that is a `struct` in a linked list. I want to access the member named `data`. I do this using the expression

a) `nodePtr.data`

[✓ b)] `nodePtr->data`

c) `*nodePtr.data`

d) data is private, you can’t access data under any circumstances.

28. Suppose `class D` is derived from `class B`, and `class B` has a `public` member function whose declaration is `void f();`. Suppose `class D` has its version of the function, `void f()`. Here is a pointer definition and an access to a member function..

```
B* bPtr = new D;
bPtr->f();
```

Suppose this is embedded in an otherwise correct and complete program. Which version of `f()` will be called?

a) `D::f()`

b) `B::f()`

c) This is illegal. You can’t assign a `D` object to a variable of type pointer to `B`.

29. A linked list is

a) fixed in size

[✓ b)] can vary in size, shrinking or growing as there is need

c) can be set, and then not changed other than destroying the list

d) none of the above

30. To determine whether a file was opened successfully, one can use the ___________ `fstream` member function

a) `close( )`

b) `overloaded operator <<( )`

c) `open( )`

d) `eof( )`

e) `fail( )`

~~31. The _______ member function takes a single `char` value from the input file, without regard to whether it is whitespace.~~

~~a) `newLine`~~

~~b) `get`~~

~~c) `put`~~

~~d) `getline`~~

~~e) `putline`~~

32. Which statement is **not true** about a pure virtual function. A pure virtual function is a member function

a) Whose declaration ends with = 0.

b) That is used in a base class

[✓ c)] Takes no arguments

d) Member form that is used to force all derived classes to implement that member function or be a pure virtual function member of the derived class.

33. How many type parameters may a function template have?

a) none, that is not what the parameters in a function template are called.

b) 1

c) 2

[✓ d)] as many as are needed

~~34. Which of the following are correct ways to end a loop using a test for end-of-file?~~

~~a)~~
```
while(inStream->next)
{
    cout << next;
```

```
  }
```

b)
```
while(inStream >> next)
  cout << next;
```

c)
```
inStream.get(next)
while(!eof(inStream))
{
  cout << next;
  inStream.get(next);
}
```

d) None of the above. You cannot control a loop using a test for end of file.

~~35. The _____________ `fstream` member function opens a file stream and connects the stream variable to a physical file whose name is the argument to the function.~~

~~a) `close( )`~~

~~b) `overloaded operator <<( )`~~

~~c) `open( )`~~

~~d) `eof( )`~~

~~e) `flush( )`~~