CS 162 Exam I Spring 2018 **FORM 1**

Please put your name and form number on the scantron.

True (A)/False (B) (28 pts, 2 pts each)

1. The assignment operator may not be used with objects of a class.
2. A struct variable is declared differently from a predefined type such as an int.
3. A function may return a struct.
4. All constructors for a class must be private.
5. The expression **s->m;** indicates that **s** is a structure pointer and **m** is a structure member.
6. File output may be formatted the same way as console screen output.
7. A destructor function can have zero to many parameters.
8. In object-oriented programming, the object encapsulates both the data and the functions that operate on the data.
9. You must, according to C++, use the **private** access specification for all data members of a class.
10. A static member function does not need to be called by a specific object of the class.
11. The overloaded = operator copies data from one object to another so it is known as the copy destructor.
12. It is legal to call a constructor as a member function of an object of a class using the dot operator, as in w.A(2).
13. In a shallow copy, pointers are followed and data and the pointer structure are duplicated.
14. C++ allows you to separate class declaration from implementation.

Multiple Choice (72 pts, 3 pts each)

```
1. #ifndef SONG_H
2. #define SONG_H
3. #include <string>
4. using namespace std;
5. struct song {
6.     string name;
7.     string artist;
8.     float length; //in minutes
9.     string genre;
10.    };
11.    #endif
```

Figure 1

15. Figure 1 would be found in a(n):
a. Implementation file
b. Driver file
c. Interface file
d. None of the above

16. What is the purpose of lines 1, 2, and 11 in Figure 1?
a. To prevent multiple includes of an interface file.
b. To let the programmer know what file they are in.
c. To take up space.
d. They indicate that the file is an interface file.

```
1. #ifndef PLAYLIST_H
2. #define PLAYLIST_H
3. #include "song.h"
4. using namespace std;
5. class Playlist {
6.    public:
7.          Playlist();
8.          void set_num_songs(int);
9.          int get_num_songs() const;
10.         void set_playlist_name(string);
11.         string get_playlist_name() const;
12.         song get_song(int) const;
13.         void add_song(song);
14.         void remove_song(int);
15.         void calculate_length();
16.         float get_length() const;
17.         Playlist(const Playlist &);
18.         const Playlist& operator=(const Playlist &);
19.         ~Playlist();
20.   private:
21.         song* list;
22.         int num_songs;
23.         string name;
24.         float length;
25.   };
26.   #endif
```

Figure 2

17. In Figure 2, why are quotes used on line 3 instead of angle brackets?

a. To indicate that the file will be found locally.

b. To confuse the programmer.

c. There is no difference between quotes and angle brackets in a #include.

d. None of the above

18. The key differences between a struct and class in C++ (the way it is used in CS 162 at Oregon State University) are:

a. There are no differences.

b. Classes are default private and nothing else.

c. Classes are default private and have functionality where structs are default public and don't have functionality.

d. Both classes and structs have functionality but classes are default public and structs are default private.

19. In Figure 2, an example of a function which destroys the object when it goes out of scope is line:

a. 7
b. 14
c. 17
d. 19

20. In Figure 2, what is line 17?

a. A non default constructor
b. A default constructor
c. A copy constructor
d. A mutator

21. In Figure 2, lines 9, 11, 12, and 16 are examples of:

a. Mutator functions
b. Accessor functions
c. Constructors
d. Generic functions

22. In Figure 2, why are lines 9, 11, 12 and 16 terminated with a const?

a. const means the function will not change the member variable.
b. const is an indicator to the programmer that the function consistent.
c. Functions that return things should be const.
d. None of the above.

```
1. Playlist() {
2.     song* list = NULL;
3.     num_songs = 0;
4.     name = "New playlist";
5.     length = 0.0;
6. }
```

Figure 3

23. There are two things wrong with Figure 3. Which lines are incorrect and for what reason? (Reference Figure 2 for class prototypes).

a. 1 is not named correctly and 5 is a mismatch type
b. 3 and 4 are mismatch types
c. 1 is missing the scope resolution operator and 2 is redeclaring the list
d.2 cannot be NULL and 6 is missing a semicolon

```
1. Playlist::Playlist(const Playlist & copy) {
2.     num_songs = copy.num_songs;
3.     name = copy.name;
4.     length = copy.length;
5.     list = new song[num_songs];
6.     if(list != NULL) {
7.           delete [] list;
8.     }
9.     for(int i=0; i<num_songs; i++) {
10.          list[i] = copy.list[i];
11.    }
12. }
13. const Playlist& Playlist::operator=(const Playlist & copy) {
14.    num_songs = copy.num_songs;
15.    name = copy.name;
16.    length = copy.length;
17.    if(list != NULL) {
18.          delete [] list;
19.    }
20.    if(num_songs == 0) {
21.          list = NULL;
22.    }
23.    else {
24.          list = new song[num_songs];
25.          for(int i=0; i<num_songs; i++) {
26.               list[i] = copy.list[i];
27.          }
28.    }
29.    return *this;
30. }
```

Figure 4

24. What are the two functions represented in Figure 4?

a. Construct and Nondefault constructor
b. Copy Construct and Assignment Operator Overload
c. Destructor and Copy Constructor
d. None of the above

25. There is an error in the first function in Figure 4. What is it?

a. The parameter should be a pass by value.
b. It should not delete the list because the object being created does not have memory allocated to it
c. The this pointer should be used in place of copy
d. Line 5 should be after lines 6-8 to delete the old list.

26. A _____ is a member function that is automatically called when a class object is _____.
    a. Constructor, created
    b. Destructor, created
    c. Static function, deallocated
    d. Utility function, declared

27. When a member function is defined outside of the class declaration, the function name must be qualified with the
    a. Class name, followed by a semicolon
    b. Name of the first object
    c. Class name, followed by the scope resolution operator
    d. Private access specifier

28. If you do not furnish a(n) _____, an automatic shallow copy will be performed when one object is assigned to another object.
    a. Overloaded constructor function
    b. Overloaded assignment operator
    c. Default constructor function
    d. Overloaded copy operator

29. A(n) _____ is a special function that is called whenever a new object is created and initialized with another object's data.
    a. Static function
    b. Destructor
    c. Copy constructor
    d. Assignment function

30. If you do not furnish a _____, a default one will be provided by the compiler.
    a. Constructor
    b. Destructor
    c. Copy constructors
    d. All of these

```
1. void print_playlist(const Playlist &p) {
2.     cout << "Playlist: " << p.get_playlist_name() << endl;
3.     cout << "Length: " << p.get_length() << " minutes" << endl;
4.     song s;
5.     for(int i=0; i<p.get_num_songs(); i++) {
6.        s = p.get_song(i);
7.        cout << i+1 << ". " << s.name << " by " << s.artist << " ";
8.        cout << s.length << " " << s.genre << endl;
9.     }
10. }
```

Figure 5

31. Figure 5 is defined and used in the driver file. Reference Figures 1 and 2 for all other appropriate prototypes. Assume that const is removed from Figure 2. Will the function in Figure 5 compile? Why or why not?
    a. Yes because the accessors do not change anything about the object.
    b. No because the compiler cannot tell if the accessors will change anything about the object.
    c. No because you can't pass by a constant reference.
    d. Yes because everything is perfectly legal but there will be a logic error.

```
1. void pop_from_file(Playlist& p, ifstream& f);
2. void print_playlist(const Playlist &p);
3. void name_the_playlist(Playlist & p);
4. song pop_song(string name,  string artist, float len, string genre);
5.
6. int main() {
7.    Playlist rock_the_test;
8.    ifstream rf;
9.    name_the_playlist(rock_the_test);
10.   rf.open("song_list.txt");
11.   if(rf.is_open()) {
12.         pop_from_file(rock_the_test, rf);
13.   }
14.   else {
15.         cout << "The file did not open" << endl;
16.   }
17.   rf.close();
18.   print_playlist(rock_the_test);
19.   Playlist triumph = rock_the_test;
20.   name_the_playlist(triumph);
21.   triumph.remove_song(0);
22.   triumph.remove_song(1);
23.   song s = pop_song("We Are the Champions", "Queen", 2.59, "Rock");
24.   triumph.add_song(s);
25.   s = pop_song("Tubthumping", "Chumbawamba", 4.38, "Dance Rock");
26.   triumph.add_song(s);
27.   print_playlist(triumph);
28.   return 0;
29.   }
```

Figure 6

For the remaining questions until the Extra Credit section, assume a correctly coded program for anything that is not shown in Figure 6. Assume that the prototypes presented have correct definitions.

32. What function is being called on line 7?
    a. Default Constructor
    b. Non default constructor
    c. Destructor
    d. Copy Constructor

33. What is the set mode of the object on line 8?
    a. Write only
    b. Append only
    c. Read only
    d. Truncate

34. Which of the Big 3 are called on line 12?
    a. Copy constructor
    b. Destructor
    c. Assignment Operator Overload
    d. None

35. Which of the Big 3 are called on line 18?
    a. Copy constructor
    b. Destructor
    c. Assignment Operator Overload
    d. None

36. Which of the Big 3 are called on line 19?
    a. Copy constructor
    b. Destructor
    c. Assignment Operator Overload
    d. None

37. What form of copy is being employed on line 23 and 25?
    a. Deep copy
    b. Shallow copy
    c. Mystery copy
    d. None

38. Is the destructor called, if so when?
    a. No
    b. Yes at the end of the copy constructor
    c. Yes at the end of the program
    d. Yes when the file was closed

39. To pass an object of class `person` to a function as a formal value parameter most efficiently you should use:
    a. `person p`
    b. `const person p`
    c. `person &p`
    d. `const person &p`

40. Given the class definition:

```
class CreateDestroy {
   public:
        CreateDestroy() { cout << "constructor called, "; }
        ~CreateDestroy() { cout << "destructor called, "; }
};
```

What will the following program output?

```
int main() {
   CreateDestroy  c1;
   CreateDestroy  c2;
   return 0;
}
```

a. constructor called, destructor called, constructor called, destructor called,
b. constructor called, destructor called,
c. constructor called, constructor called,
d. constructor called, constructor called, destructor called, destructor called,

41. True (A) or False (B): the official C++ term for a function in a class which alters the value of a member variable is manipulator.
42. True (A) of False (B): comparing two variables of a struct type with the == will, through default behavior, return true if each member variable of the two are the same (ie if (circle1 == circle 2)).