Midterm Exam #2, Fall 2018, CS 141: Programming Design II, Prof. Reed

This test is worth 15% of your final grade. This test is open notes (anything on paper), but no electronic resources may be used. When only a program segment is given, you can assume it is otherwise placed in the context of a program that otherwise works correctly.

For the multiple-choice questions with a bubble next to the answers, you must fill in only one bubble next to the best answer. Some questions have an indicated rectangle where you must write down your answer. Any writing outside of these indicated areas will not count towards your grade.

There are 19 multiple-choice problems worth 3 points each (19 x 3pts. = 57pts), and there are 7 free-response problems worth 6 points each (7 x 6pts. = 42 pts), + 1 point for free = 100 total.

1. What is the result of executing the following program statement, which is part of an otherwise correct program?
   
   ```
   cout << INT_MAX + 1;
   ```

   ○ It causes a compiler error
   ○ It displays the largest positive integer
   ○ It displays an error message
   ○ It displays a negative number

2. Consider the two function definitions shown below:

   ```
   int multiply( int c, int d)
   {
      return c * d;
   }

   int multiply( int a, int b, int c=0)
   {
      return a * b * c;
   }
   ```

   What would be the result of attempting to compile and run this code when calling it with:
   
   ```
   cout << multiply(2, 3) << endl;
   ```

   ○ It would not compile correctly
   ○ It would compile correctly, however at run time would give an error
   ○ It would compile correctly and run correctly in some cases but not in others
   ○ It would compile correctly and run correctly

3. Consider the following function declaration:

   ```
   void f1( char c = 'a', float pi = 3.14, bool b = true);
   ```

   How many of the function calls shown at right below would match the above declaration?

   ○ 1
   ○ 2
   ○ 3
   ○ 4

   ```
   I. f1( )
   II. f1( 't')
   III. f1( 'e', 9.5)
   IV. f1( 's', 9.5, false)
   ```
4. Consider trying to find a particular number within an **unsorted** array of 100 unique random numbers. If you had to repeat this type of search hundreds of times, on average how many numbers will have to be examined before the number you are searching for is found? Select the range below that includes this number.
   - 0 to 4
   - 5 or 6
   - 7 or 8
   - 9 or 10
   - 11 to 49
   - 50 to 100

5. Consider using **binary search** to find a particular number within a **sorted** array of 100 unique random numbers. If you had to repeat this type of search hundreds of times, on average how many numbers will have to be examined before the number you are searching for is found?
   - 0 to 4
   - 5 or 6
   - 7 or 8
   - 9 or 10
   - 11 to 49
   - 50 to 100

6. Consider the following array of numbers to be sorted:
   
   \[5 \ 2 \ 4 \ 3 \ 1\]

   Assume we are sorting into ascending order, and we always start our passes at the right-hand side. A comparison costs one point, and a swap costs two points. How many points does **Bubble Sort** take, using the process we discussed in class? Write down your answer in the rectangle at right above. Print legibly.

7. Consider the following array of numbers to be sorted:
   
   \[5 \ 2 \ 4 \ 3 \ 1\]

   Assume we are sorting into ascending order, and we start with the element at the left-hand side. A comparison costs one point, and a swap costs two points. How many points does **Selection Sort** take, using the process we discussed in class? Write down your answer in the rectangle at right above. Print legibly.
8. Assume we are given the following function definition:

```c
void f8( char *pString1, char *pString2)
{
    char *pTemp = strchr( pString2, '_');
    *pTemp = '\0';
    strcat( pString1, pString2);
    cout << pString1 << endl;
}
```

What is the output of the code segment below that uses the above function? Write down your answer in the rectangle at right. Print legibly.

```c
char string1[ 81] = "to_be_";
char string2[ 81] = "or_not_";
f8( string1, string2);
```

9. Assume we are given the following function definition:

```c
void f9( char *pWord, char a, char b)
{
    char *pTemp = pWord;
    while( pTemp) {
        pTemp = strrchr( pWord, a);
        if( pTemp) {
            *pTemp = b;
        }
    }
}
```

What is the output of the code segment below that uses the above function? Write down your answer in the rectangle at right. Print legibly.

```c
char string3[ 81] = "hee hee"

f9( string3, 'e','a');
cout << string3;
```

10. Consider the following function declaration:

```c
void displayTable( int values[ ][ ]);
```

Note that there is no value inside either the first or second set of square brackets to specify the array size. Which of the following statements is the best description of this situation?

- ( ) Values are not required in either set of brackets
- ( ) You must have a value in the first set of brackets, but not necessarily the second
- ( ) You don’t need a value in the first set of brackets, but do in the second
- ( ) You must supply values for both sets of brackets
11. Consider the following function declaration:

```cpp
void setValues( int values[ ] );
```

Note that there is no value inside the square brackets to specify the array size. Which of the following statements is the best description of this situation?

- This will cause a compiler error and will not run
- This will compile, but will cause a run-time error
- Having no number will allow writing past the end of the array, however this would not be possible if a number value is supplied.
- Having no number will allow writing past the end of the array, and this would still be possible even if a number value were supplied.

12. When calling a class `get` member function, what is the effect of adding `const` to the end of the function header?

- Changes to data members have scope only within the member function
- Changes to data members can be made. The `const` serves as documentation only.
- The compiler will generate an error message if a data member change is attempted
- An attempt to make a data member change results in a run-time error

13. Which of the following is the best description of objects and classes?

- Classes are a general category, while objects are particular instances.
- Objects are a general category, while classes are particular instances.
- Both objects and classes are synonyms that describe a general category
- Both objects and classes are synonyms that describe particular instances

14. Imagine we create a Date class with the following single constructor:

```cpp
Date( int theMonth, int theDay, int theYear ) {
    month = theMonth; day = theDay; year = theYear;
}
```

The driver code then has the following, which generates an error:

```cpp
Date d1;
```

What is the likely reason for this error?

- The Date class is declared after the declaration of d1
- There is already a declared Date object with the name d1
- This is a declaration and is not calling the Date class constructor
- There is no default constructor for the Date class

15. Imagine we are using one of the later versions of the Date class that we developed.

The driver code then has the following, which when compiled gives a warning:

```cpp
Date d2();
```

What is the likely reason for this warning?

- The compiler understands this as a function declaration rather than a call to the Date class constructor
- There is no default constructor for the Date class
- d2 is itself a type or variable already defined within the program
- The parenthesis should be after Date instead of being after d2
16. Consider the code segment shown below:

```cpp
class Date
{
    public:
        Date( int day=1, int month=2, int year=2000) {}
    private:
        int day, month, year;
};

int main()
{
    Date d1(3,4,5);
    cout << d1.day << "/" << d1.month << "/" << d1.year << endl;
    return 0;
}
```

What is the output of the above code segment?
- It does not compile, so there is no output.
- It compiles, but gives a run-time error when run.
- 1/2/2000
- 3/4/5

17. Imagine we create a Date class with private data members for month, day and year. How many of the three set member functions defined inside the Date class shown below would compile and run?

```cpp
class Date
{
    public:
        // other code here
        // ...
        void setMonth( int month) {
            month = month;
        }
        void setDay( int day) {
            Date::day = day;
        }
        void setYear( int year) {
            this->year = year;
        }
    private:
        int month, day, year;
};
```

- 0
- 1
- 2
- 3
18. In class we saw an example where Employee e2 was created and initialized as a copy of Employee e1. When we later changed the startDate for e2, the startDate for e1 ended up being changed as well. How were we able to fix this?
- We had the Employee copy constructor call the Date copy constructor
- We had the Employee copy constructor make a new Date and used it with the Date copy constructor
- It was not possible to fix this without initializing e2 as a separate Employee rather than a copy of e1
- The two start dates were made different from each other by initializing them to different values one after the other.

19. What is output from running the program segment shown below at left? Write down your answer in the box shown at right below. Print legibly.

```cpp
int age = 19;
if (20 <= age <= 30) {
    cout << "Yes";
} else {
    cout << "No";
}
```

20. Consider the code segment shown at left below. What is the output when function scope() is called? Write down your answer in the rectangle at right below. Print legibly.

```cpp
int y = 4; // global variable
void s1(int x)
{
    cout << x+y << endl;
}
void s2(int x)
{
    x = y++;
    s1(y);
}
void scope()
{
    int x = 1;
    s2(x);
}
```
21. Consider the code shown at left below. What is the output of the following C++ program segment, called with `confuseDriver()`?
Write down your answer in the box shown at right below. Print legibly.

```
int s=1, z=3;

void confuse1(int a, int s)
{
    s++;
    a++;
}

void confuse2(int b, int &s)
{
    z = b++;
    s = b + 3;
}

void confuse3(int &a, int &s)
{
    a = s + 1;
    s++;
}

void confuseDriver()
{
    int z=0;

    confuse1(s, z);
    confuse2(s, z);
    confuse3(s, z);
    cout << s+z << endl;
}
```

22. Consider the code provided for program 4 that was reviewed in class. How are the colors for the squares on the playing board determined?
   ○ We use the provided `getRandomColor()` function, which uses `srand()` to always give different results
   ○ We use the provided `getRandomColor()` function, which always gives the same order of colors
   ○ We rotate through a provided list of colors
   ○ We read the color settings from the provided file, allowing more convenient testing

23. For program 4 as discussed in class, consider the situation where we have a same-color Square adjacent to the upper-left square when the board is created. What problem is caused if this adjacent same-color Square is not marked as part of the captured area when the board is created?
   ○ If not originally marked, its color never changes.
   ○ If not originally marked, it won't change color on a first move when it should
   ○ If not originally marked, it will always change color on a first move even when it shouldn't
   ○ It will always change color on a second move, even when it shouldn't
24. For program 4 as discussed in class, what pieces of information must the program consider in deciding whether or not to advance to the next level?
   - Only how many pieces on the current board are the same color
   - Only how many moves have been made
   - How many pieces on the board are the same color, and how many moves have been made
   - How many pieces on the board are the same color, how many moves have been made, and the size of the board.

25. For program 4 as discussed in class, what approach was recommended to store the array of Square objects?
   - Declare an array using a variable, so that when the variable changes, the size of the array changes.
   - Declare an array using a Vector, which lends itself to growing dynamically
   - Declare an array of Square pointers, which gives flexibility of later allocating the Square objects themselves.
   - Declare an array of Square objects to be the maximum size the program ever uses

26. Consider the code provided for program 4 that was reviewed in class. Why is it necessary to add an extra boolean variable to check whether or not the mouse has already been clicked?
   - Without that variable each additional square added to the captured area incorrectly causes the move number to be incremented
   - That variable is necessary to eventually exit the event loop
   - Without that variable the move number will be double what it should be, being incremented both on the mouse down and mouse up events
   - Without that variable the move number will be hundreds of times greater than it should be because of event loop repetitions