1. (**15 points**) There are six logic or syntax errors in the following program; find *five* of them. Circle each of the five errors you find and write the line number and correction in the space provided below.

```
 1 #include "iostream"
 2 using namespace std;
 3
 4 void main() {
 5
 6   cout << "This program computes Fn, the nth fibonacci number."
 7        << endl;
 8
 9   do {
10     cout << "Enter a positive integer: ";
11     cin >> n;
12   } while (n < 1);
13
14   if (n = 1 || n == 2)
15     cout << "F" << n << " = 1" << endl;
16   else {
17     int a = 1, b = 1;
18     for (int i = 3; i <= n; i++)
19       int c = b;
20       b + a;
21       a = c;
22     cout << "F" << n << " = " << b << endl;
23   }
24
25   return 0;
26 }
```

Line Number	Correction
1	Change "iostream" to <iostream>
4	Change void main() to int main()
11	n not declared; add declaration, e.g. on line 5
14	Change n=1 to n ==1
18 / 21	Need { } around body of for loop
20	Change b+a to b = b + a or b+= a

2. (**18 points**) Complete the code:

   a. Complete the following interface for a Dinosaur class:

```
class Dinosaur {
```

public:

```
  void SetDescription(string description);
  void SetCarnivorous(bool carnivorous);
  string GetDescription() const;
  bool GetCarnivorous() const;
```

private:

```
  string m_description; // description
  bool m_carnivorous;   // carnivorous if true
```

}; **must have semicolon**

   b. Complete the implementation of the `GetDescription()` method of the `Dinosaur` class. Assume the implementation is in a separate file from the interface.

```
string
```

Dinosaur::GetDescription

```
() const {
```

return m_description;

```
}
```

   c. In my main function, I create a `Dinosaur` object named `barney` and use the "Set" methods to initialize `barney`'s variables. Complete the following code to print `barney`'s description:

```
cout <<
```

barney.GetDescription()

```
<< endl;
```

3. (**2 points**) Why are the two "Get" functions in problem (2) `const`?

They are accessor methods that do not modify any of the class variables.

Do not have to use term "accessor"; must say they do not modify class variables.

4. (**10 points**) Will the following program work as the programmer intended? Why or why not? Address both the method by which parameters are passed *and* the logic of the `UpperCase()` function.

```
1    #include <iostream>
2    using namespace std;
3    void UpperCase(char str[]);
4    int main() {
5      char msg[80] = "I'm having candy for dinner!";
6      // Convert msg[] to upper case and print it to the screen
7      UpperCase(msg);
8      cout << msg << endl;
9    }
10   void UpperCase(char str[]){
11     for (int i = 0; str[i] != '\0'; i++)
12       if ( 'a' <= str[i] && 'z' >= str[i] )
13         str[i] = str[i] - 'a' + 'A';
14   }
```

Yes, it will work as expected.

C-string is passed as address (reference), so the changes made in UpperCase() will be visible in main()

Have to say something like C-string / arrays are passed as pointers or passed by reference.

The function UpperCase() uses character literals to determine if a letter is lower case and to covert it to upper case

Reasonable description of the function logic and indication that they understand the use of char literals.

5. (**10 points**) What will the following program print to the screen? Complete the boxes below and explain your answers.

```
 1 #include <iostream>
 2 using namespace std;
 3 int main() {
 4   double data[5] = {-1.0, 3.14, 7.59, -0.3, 1.1};
 5   double max = data[0];
 6   int maxi = 0;
 7   // Find the maximum value and index of the maximum value
 8   // in the data array
 9   for (int i = 1; i < 5; i++) {
10     if (data[i] > max) {
11       int max = data[i];
12       maxi = i;
13     }
14   }
15   cout << "max value is " << max << endl;
16   cout << "max occurs at index " << maxi << endl;
17 }
```

Output:

max value is **-1.0**

max occurs at index **4**

Explanation:

Because the variable max is re-declared within the body of the if statement, it's scope is the body of the if. The modification to max on line 11 is limited to the scope of the if statement and so when max is printed on line 15, the original value from line 5 is printed (data[0] which is -1.0).

The variable maxi is not re-declared within the body of the if statement, so the modification on line 12 effects the variable in the main() scope. The value of maxi increases so long as data[i] > max, and max is *always* -1.0, so the end value of maxi is 4, the last index of the array.

Must demonstrate understanding of scope

6. I want to write a function `RemoveSpaces()` to remove the spaces from a C-string. The C-string to be processed will be passed as a parameter, and the resulting string, with spaces removed, should overwrite the original string. The function has no return value.

   a. (**8 points**) Complete the function prototype:

void `RemoveSpaces(` **char str[]** `);`

   b. (**12 points**) Write the implementation of the `RemoveSpaces()` function:

```
void RemoveSpaces( char str[] ) {

      int i, j;

      for (i = 0, j = 0; str[i] != '\0'; i++)

            if (str[i] != ' ')

                  str[j++] = str[i];

            }

      str[j] = '\0';

      return;
}
```

7. (**6 points**) Write a function header comment for your `RemoveSpaces()` function from (6.b), including a description of the function, its pre-conditions, and post-conditions.

```
RemoveSpaces() - remove the spaces from a C-string
Pre-conditions: str is a null-terminated C-string
Post-conditions: spaces are removed from the input string;  the
transformed string overwrites the original string.  The transformed
string will be null-terminated.
```

8. (4 points) *Encapsulation* is an important concept in object-oriented programming.

   a. Define *encapsulation:*

Encapsulation is the bundling of data and methods (variables and functions) along with methods of restricting access to data and methods.

   b. Give an example of how encapsulation is implemented in C++:

C++ classes combine data and methods and provide mechanisms for restricting access to both.

9. (**15 points**) Consider the following program:

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 int main() {
 5
 6   char word[] = "terryology";
 7
 8   char *cptr = &word[9];
 9
10   while (cptr >= word) {
11     cout << *cptr;
12     cptr--;
13   }
14   cout << endl;
15 }
```

a. What is the output of the program?

ygoloyrret

b. Describe the values that are compared in the conditional `cptr >= word` on line #10 in terms of the elements of the `word[]` array.

cptr is the address of a letter in word[]; word is the address of the first letter in word[]. Therefore, the comparison tests whether cptr is pointing to a letter after the first letter.

c. Explain the different meanings of the "`*`" character on line #8 and line #11.

Line 8: * is part of the declaration of cptr; it says the cptr is a pointer variable.
Line 11: * is a pointer dereference; *cptr is the char pointed to by cptr.

Page	Points	Earned
1	15	
2	20	
3	10	
4	10	
5	20	
6	10	
7	15	
Total	**100**	