

1. (18 points) There are six logic or syntax errors in the following program. Circle each error and write the line number and correction in the space provided below.

```

1   #include <iostream.h>
2   using namespace std;
3   int main() {
4       int n = -1;
5       int fact;
6       cout << "This program computes n factorial."
           << endl;
7       do {
8           cout << "Enter a positive integer n: ";
9           cin << n;
10          } while (n < 0);
11         if (n = 0)
12             cout << "Factorial of 0 is 1" << endl;
13         else
14             for (int i = 1; i < n; ++i)
15                 fact *= i;
16         cout << "Factorial of " << n << " is " << fact
           << endl;
17         return 0;
18     }
```

+1 pt each correct line number, +2 points each correct fix

Line Number	Correction
1	#include <iostream> (no .h)
5	int fact = 1;
9	cin >> n;
11	if (n == 0)
14	for (int i = 1; i <= n; i++)
16	cout << "Factorial of " << n << " is " << fact << endl;

2. (8 points) Complete the code:

- a. I want to compute the *average* of the two integer variables *x* and *y* and save it to the double variable *avg*.

```
avg = ( x + y ) / 2.0 ;    +2 points
```

- b. A race of aliens from a planet with a six hour day wants to convert 11 am local earth time to the time on their home planet:

```
int umbcTime = 11;
int alientTime;
alientTime = umbcTime % 6;    +2 points
```

- c. The program should only call the function *ReturnGrades()* if the variable *numStudents* has a value between 1 and 500, inclusive:

```
if ( numStudents >= 1 && numStudents <= 500 )    +2 points
    ReturnGrades(grades, numStudents);
```

- d. The user of a data analysis program can enter 's' to save their data or 'h' to display a help message. The users selection is stored in the variable *selection*:

```
switch( selection ) {
    case 's':
        SaveData();
        break;
    case 'h':
        DisplayHelp();
        break;    +2 points
    default:
        cerr << "Invalid selection" << endl;
}
```

3. (8 points) Explain why the following program will not do what the programmer intended:

```
1  #include <iostream>
2  using namespace std;
3  void AbsValue(double x);
4  int main() {
5      double x = -7.251;
6      // Replace x with its absolute value
7      AbsValue(x);
8  }
9  void AbsValue(double x) {
10     if ( x < 0.0 )
11         x = -x;
12 }
```

The argument x is passed by value, so although the parameter x is changed within `AbsValue()`, there is no change to the value in `main()`.

+4 points "passed by value", +2 if on the right track

+4 points "changes in function but not in main", +2 if on the right track

4. (4 points) What is the value of x in the following code sample? Circle the correct answer.

```
1  int a = 2, b = 3, c = 5, d = 8;
2  int x = a + b * d / c;
```

a. 5

c. 8

b. **6**

d. 10

+ 4 points for correct answer (b); no partial credit

5. (8 points) What will the following program print to the screen? Complete the boxes below.

```

1   #include <iostream>
2   using namespace std;
3   int main() {
4       int i = 3, j = 4;
5       {
6           int j = 5;
7           cout << i << " " << j << endl;
8           i += j;
9       }
10      cout << i << " " << j << endl;
11  }
```

Output:

+2 points for each correct answer

3	5
8	4

6. (6 points) List the names of the *arguments* and the *parameters* in the following example code:

```

1   #include <iostream>
2   using namespace std;
4   int Sum(int x, int y, int z);
5   int main() {
6       int a = 1, b = 2, c
7       cout << Sum(a, b, c) << endl;
8   }
9   int Sum(int x, int y, int z) {
10      return x + y + z;
11  }
```

+1 point each correct answer

Arguments:

a, b, c

Parameters:

x, y, z

7. The *trace* of a matrix is the sum of its diagonal entries. I want to write overloaded functions to compute the trace of a double or integer matrix, stored as a two-dimensional array. The trace of an integer array is an integer, and the trace of a double array is a double. The array will be declared to have MAX_SIZE rows and columns, but the actual size will be passed as an integer argument to the function.

a. (8 points) Complete the function prototypes for the two functions:

```
int Trace(int array[][MAX_SIZE], int size);
```

```
double Trace(double array[][MAX_SIZE], int size);
```

+1 for each return type, +2 for each array parameter, +1 for each int array size

b. (12 points) Write the double version of the *Trace()* function:

```
double Trace(double array[][MAX_SIZE], int size) {  
  
    double tr = 0.0;                                +3 initialize variable  
  
    if (size <= MAX_SIZE) {                          +1 check value of size (*)  
        for (int i = 0; i < size; ++i) {            +3 loop syntax  
            tr += array[i][i];                      +3 sum correct element  
        }  
    }  
  
    return tr;                                       +2 return value  
}
```

8. (12 points) Write a function header comment for your *Trace()* function from (7.b), including a description of the function, its pre-conditions, and post-conditions.

```
/*
 * Trace() - compute the trace of a double matrix +3 function name
 *
 * Preconditions
 *   The size-by-size sub-array of array contains +3 precon. #1
 *   valid data.
 *   size is  $\geq 1$  and  $\leq \text{MAX\_SIZE}$  +3 precon. #2
 * Postconditions
 *   returns the trace of the size-by-size matrix +3 postcon.
 */
```

9. (8 points) We've learned about three different loop statements. In each of the following situations, which is the *most* appropriate?

- a. A user will be prompted to enter an integer in the range one to ten, inclusive; the prompt will be repeated until the user enters a value in the correct range.

do-while **+2 points**

- b. Sum the values in a fixed-length double array.

for **+2 points**

- c. If there is a data file in a particular directory, read the file and process the data; repeat so long as there are still data files in the directory.

while **+2 points**

- d. At the end of the semester, compute the lab average for a particular student.

for **+2 points**

10. (8 points) What output is produced by the following code?

```
1   int *p1, *p2;
2   int x = 3, y = 5;
3   int z[3] = {1, 2, 3};
4   p1 = &x;
5   p2 = &y;
6   cout << *p1 * *p2 << endl;
7   p2 = p1;
8   cout << *p1 + *p2 << endl;
9   p1 = z;
10  cout << *(p1+1) * *p2 << endl;
11  cout << *(p1+2) / *p2 << endl;
```

+2 points for each correct answer

15
6
6
1

Page	Points	Earned
1	18	
2	8	
3	12	
4	14	
5	20	
6	20	
7	8	
Total	100	