COSC 1560 – Computer Programming II

Mid-Term Examination #1

NAME:

Attempt all questions. The number of points for each question is shown. The total is 100 points.

[16 points]

1) The function 'lowest' should find the alphabetically lowest character in an array. Identify syntax ("*compilation*") errors in the following code, and show how they should be corrected. *Write the corrected code as a comment on the same line.*

```
#include <iostream>                          //
using namespace std;                         //

char lowest(char values[], int size)         // char lowest(const char values[], int size);
int main()                                   //
{
        int n = 3;                           // const int n=3;
        char letters[n] = {A, H, M};         // char letters[n] = {'A', 'H', 'M'};
        char result;                         //
        result = lowest(letters, n);         //
        cout << "Result:  " << result << endl; //
        return 0;                            //
}

lowest(char values[], int size);             // char lowest(const char values[], int size)
{
        int result = values[];               // char result = values[0];
        for (i=1; i < size; ++i)             // for (int i=0; i < size; ++i)
        {
                if (values[i] < result)      //
                {
                        result = values[i];  //
                }
        }
        return result;                       //
}
```

[8 points]

2) Determine the output in each 'cout' statement in the following code:

```
void f(int a, int& b)        // 'a' passed by "value", 'b' passed by "reference"
{                            // a = 10, b = 15
        ++a;                 // a = 11
        b++;                 // b = 16
        cout << a << endl;               // 11
        cout << b << endl;               // 16
}
int main()
{
        int x = 10;
        int y = 15;
        f(x, y);             // x will not be updated, y will be updated since it
                             // is passed by "reference"
        cout << x << endl;               // 10
        cout << y << endl;               // 16
        return 0;
}
```

[16 points]

3) A file named "Data.txt" contains the following six numbers:

10.5 12.3 11.6 9.0 8.7 10.5

Write code to open the file for reading, determine and display the smallest number, and close the file.

```
ifstream dataFile;
dataFile.open("Data.txt");
double num;
double smallest;
dataFile >> smallest;
for (int i=0; i < 5; ++i) {
        dataFile >> num;
        if (num < smallest) {
                smallest = num;
        }
}
cout << "Smallest is: " << smallest << endl;
dataFile.close();
```

[8 points]

4) The following array is to be sorted into ascending order using the 'bubble sort' algorithm:

8	5	1	9	7

The first iteration of the algorithm results in the largest value being moved to the end of the array. Use the following table to show the order of the values after this first iteration.

5	1	8	7	9

The bubble sort algorithm starts by comparing the first two elements (8 and 5) and swaps them if they are no in order. In this case it will swap 8 and 5. Then it will compare the next two elements (8 and 1). These will also be swapped. Then it will compare the next two elements (8 and 9). These will not be swapped since they are in order. Then it will compare the next two elements (9 and 7). These will be swapped. We are then at the end of the array so the first iteration has been completed and 9 is at the end of the array, which is its final position.

The algorithm will then do the same process on the first 4 elements, which will result in 8 being moved to element 4. etc.

[16 points]

5) The 'linear search' function, to search an array of 'int', is drafted below. Provide the appropriate code to replace each "XXX".

```
int linearSearch(const int list[], int numElems, int value)
{
        int index = 0;
        int position = -1;
        bool found = false;
        while ((index < numElems) && !found) {
                if (XXX) {                      // if (list[index] == value) {
                        found = true;
                        position = XXX;         // position = index;
                }
                index++;
        }
        return XXX;                             // return position;
}
```

[18 points]

6) A function named "tester" takes an array of 'double' as the first argument, the number of elements in the array as the second argument, and a value of type 'double' as the third argument. The function should return the number of elements in the array that when multiplied by the third argument are larger than 100. Write the function.

```
int tester(const double numbers[], int numElements, double value)
{
        int largeValues = 0;
        for (int i=0; i < numElements; ++i) {
                if (numbers[i]*value > 100) {
                        largeValues++;
                }
        }
        return largeValues;
}
```

[18 points]

7) The following is an array of 5 integers:

```
int values[] = {5,8,1,2,9};
```

5	8	1	2	9
1200	1204	1208	1212	1216

Indicate the output of the following lines of code:

```
int* ptr = &values[2];          // 'ptr' is assigned to the 3rd element of the array
                                // So 'ptr' is assigned 1208, so 'points' to 1208

cout << *ptr << endl;           // 1
                                // "The value at the location to it points"

cout << ptr << endl;            // 1208
                                // The value of 'ptr'

cout << (*(ptr – 2)) * (*ptr + 1) << endl;      // 5 * 2 = 10
                                // *(ptr – 2) is the value at the location that
                                // 'ptr – 2' points (5)
                                // (*ptr + 1) is the value to which 'ptr' points
                                // plus 1 (1 + 1 = 2)

--ptr;                          // 'ptr' now points to 1204

cout << *ptr + 10 << endl;                  // 18
                                            // Value at 1204 plus 10: 8+10 = 18

cout << *(ptr + 2) << endl;                 // 2
                                            // Value at which (ptr + 2) points
                                            // 'ptr + 2' is a pointer to 1212
                                            // Value at 1212 is 2

ptr = values + 3;                           // 1212
                                            // 'values' points to the first element
                                            // of the array
                                            //  So 'ptr' points to the 4th element of
                                            // the array, address 1212

cout << *ptr << endl;                       // 2
                                            // Value at 1212 is 2
```