Pointer Review Solutions

1. The subparts to this problem involve errors in the use of pointers.
    a.

```
int main()
{
   int arr[3] = { 5, 10, 15 };
   int* ptr = arr;

   *ptr = 10;          // set arr[0] to 10
   *(ptr + 1) = 20;      // set arr[1] to 20
   ptr += 2;
   ptr[0] = 30;        // set arr[2] to 30

   while (ptr >= arr) {
      cout << ' ' << *ptr;    // print values
      ptr--;
   }
   cout << endl;
   return 0;
}
```

    b.

```
void findDisorder(int arr[], int n, int* &p)
{
    for (int k = 1; k < n; k++)
    {
        if (arr[k] < arr[k-1])
        {
             p = arr + k;
             return;
        }
    }
    p = nullptr;
}

int main()
{
   int nums[6] = { 10, 20, 20, 40, 30, 50 };
   int* ptr;

   findDisorder(nums, 6, ptr);
   if (ptr == nullptr)
      cout << "The array is ordered" << endl;
   else {
        cout << "The disorder is at address " << ptr
             << endl;
        cout << "It's at index " << ptr - nums << endl;
```

```
            cout << "The item's value is " << *ptr << endl;
        }
        return 0;
    }
```

c.

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

void hypotenuse(double leg1, double leg2, double* resultPtr)
{
    *resultPtr = sqrt(leg1*leg1 + leg2*leg2);
}

int main()
{
    double x;
    double* p = &x;
    hypotenuse(1.5, 2.0, p);
    cout << "The hypotenuse is " << *p << endl;
}
```

d. The match function is supposed to return true if and only if its two C string arguments have exactly same text. Explain what the problems with the implementation of the function are, and show a way to fix them.

```
// return true if two C strings are equal
bool match(const char str1[], const char str2[]) {
    while (*str1 != 0 || *str2 != 0) { // zero bytes at ends
        if (*str1 != *str2)  // compare corresponding chars
            return false;
        str1++;            // advance to the next character
        str2++;
    }
    return *str1 == *str2;   // both ended at same time?
}

int main()
{
    char a[10] = "pointy";
    char b[10] = "pointless";

    if (match(a,b))
        cout << "They're the same!\n";
    return 0;
}
```

e. This program is supposed to write 1 4 9 16 25 36 49 64 81 100, but it probably does not. What is the program doing that is incorrect? (We're not asking you explain why the incorrect action leads to the particular outcome it does and we're not asking you to propose a fix to the problem.)

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

// Changes n, but return of arr does nothing
int* computeSquares(int& n)
{
    int arr[10];
    n = 10;
    for (int k = 0; k < n; k++)
        arr[k] = (k+1) * (k+1);
    return arr;
}

void f() // Does nothing
{
    int junk[100];
    for (int k = 0; k < 100; k++)
        junk[k] = 123400000 + k;
}

// Pointer returned is to a local variable that is already
// out of scope
int main()
{
    int m;
    int* ptr = computeSquares(m);
    f();
    for (int i = 0; i < m; i++)
        cout << ptr[i] << ' ';
    return 0;
}
```

2.

a. **string* fp;**
b. **string fish[5];**
c. **fp = &fish[4]; OR fp = fish + 4;**
d. ***fp = “tilapia”;**
e. ***(fish + 3) = “salmon”;**
f. **fp -= 3;**
g. **fp[1] = “tuna”;**

h. **fp[0] = “ono”;**
i. **bool d = (fp == fish);**
j. **bool b = (*fp == *(fp + 1));**

3.

a.

```
double computeAverage(const double* scores, int nScores)
{
    const double* ptr = scores;
    double tot = 0;
    for(int i = 0; i < nScores; i++)
    {
        tot += *(scores + i); // OR tot += *(ptr + i);
    }
    return tot/nScores;
}
```

b.

```
const char* findTheChar(const char str[], char chr)
{
    for (int k = 0; *(str + k) != 0; k++)
        if (*(str + k)== chr)
            return (str + k);

    return nullptr;
}
```

c. Now rewrite the function shown in part b so that it uses neither square brackets nor any integer variables. Your new function must not use any local variables other than the parameters.

```
const char* findTheChar(const char* str, char chr)
{
    for (const char* tmp = str; tmp != 0; tmp++)
        if (*tmp == chr)
            return tmp;

    return nullptr;
}
```

4.

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

```
// Compares two values, dereferenced by pointers
int* minimart(int* a, int* b)
{
    if (*a < *b)
        return a;
    else
        return b;
}

// Swaps two pointers locally; doesn't actually swap them
void swap1(int* a, int *b)
{
    int* temp = a;
    a = b;
    b = temp;
}

// Swaps two values, dereferencing the pointers
void swap2(int* a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main()
{
    int array[6] = { 5, 3, 4, 17, 22, 19 };

    int* ptr = minimart(array, &array[2]); // ptr = array[2]
    ptr[1] = 9; // array[3] = 9;
    ptr += 2;
    *ptr = -1; // array[4] = -1
    *(array+1) = 79; // array[1] = 79

    cout << "diff=" << &array[5] - ptr << endl; // 1

    swap1(&array[0], &array[1]); // does nothing
    swap2(array, &array[2]); // array[0] = 4; array[2] = 5

    for (int i = 0; i < 6; i++)
        cout << array[i] << endl; // 4 79 5 9 -1 19
    return 0;
}
```

5.
```
 void deleteG(char* str) {
  char* result = str;
  for (; *str != 0; str++)
       if (tolower(*str) != 'g') {
            *result = *str;
            result++;
       }
  *result = 0;
}
```

6.
```
  #include <iostream>
  using namespace std;

  void calcAvg(int* arr, int size);
  void sort(int* arr, int size);
  double calcMedian(int* arr, int size);
  int calcMode(int* arr, int size);

  int main() {

       int* students;
       double numStudents;

       cout << "How many students were surveryed? ";
       cin >> numStudents;
       cout << endl;

       while (numStudents < 0) { // Input Validation.
            cout << "The amount of students may not be a
  negative number." << endl;
            cout << "Please enter a valid number: ";
            cin >> numStudents;
            cout << endl;
       }

       students = new int[numStudents]; // Dynamically
  allocating space for the array.

  // Sets each number of movies watched to its respective
  student within the array.
       for (int i = 0; i < numStudents; ++i) {
            cout << "How many movies did student " << i + 1
  << " view? ";
            cin >> *(students + i);
```

```
          while (*(students + i) < 0) { // Input
Validation.
               cout << "The amount of movies watched may
not be a negative number." << endl;
               cout << "Please enter a valid number: ";
               cin >> *(students + i);
               cout << endl;
          }
     }

     cout << endl;
     sort(students, numStudents);
     cout << "The most common number of movies watched by
the students is(mode): " << calcMode(students, numStudents)
<< endl;

     cout.setf(ios::fixed);
     cout.setf(ios::showpoint);
     cout.precision(2);

     cout << "The median is: " << calcMedian(students,
numStudents) << endl;
     calcAvg(students, numStudents);
     cout << endl;

     delete[] students;
}

// Adds up all the values and divides it by the number of
students.
void calcAvg(int* arr, int size) {
     double sum = 0;
     for (int i = 0; i < size; ++i) {
          sum += arr[i];
     }
     cout << "The average number of movies these college
students watch a month is: "
          << sum / size;
}

// Sorts the array in ascending order.
void sort(int* arr, int size) {
     int valueHolder = arr[0];
          for (int i = 0; i < size; ++i) { // Puts array in
order here.
               for (int j = 0; j < size - 1; ++j) {
                    if (arr[j + 1] < arr[j]) {
                        valueHolder = arr[j];
                        arr[j] = arr[j + 1];
                        arr[j + 1] = valueHolder;
                    }
                }
            }
        }

        double calcMedian(int* arr, int size) {
            int middleEven = size / 2;
            int middleOdd = (size - 1) / 2;

            if (size % 2 == 0) // If number of elements is even.
                return ((*(arr + middleEven - 1) + *(arr +
        middleEven)) / 2.0);
            else // If number of elements is odd.
                return  *(arr + middleOdd);
        }

        int calcMode(int* arr, int size) {

            int maxCount = 0;
            int count = 0;
            int mode;
            for (int i = 0; i < size; ++i) {
                count++;
        //Since it is a sorted array now I am just checking is the
        consecutive positions are equal.
                if (*(arr + i) != *(arr + i + 1)) {
                    if (count > maxCount) {
                        maxCount = count;
                        mode = *(arr + i);
                        count = 0;
                    }
                }
            }
            if (maxCount == 1)
                return -1;
            else
                mode;
        }
```