Pointers Review

This project is designed to help you master pointers. To that end, you'll get the most out of it by working through the problems by hand. Only after that should you resort to running the programs (and stepping through them with the debugger) to check your understanding. Remember, on the final exam you'll have to be able to do problems like this by hand.

This "project" is more like a homework assignment, except for the last problem. There are six problems.

In problems that ask you to change code, make the few changes necessary to fix the code without changing its overall approach. For example, don't fix the program in problem 1a by changing it to

```
int main()
{
    cout << " 30 20 10" << endl;
    return 0;
}
```

1. The subparts to this problem involve errors in the use of pointers.
   a. This program is supposed to write **`30 20 10`**, but it doesn't. Find all of the bugs and show a fixed version of the program:

```
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) {
      ptr--;
      cout << ' ' << *ptr;    // print values
   }
   cout << endl;
   return 0;
}
```

   b. The findDisorder function is supposed to find the first item in an array that is less than the element preceding it, and set the p parameter to point to that item, so the caller can know the location of that item. Explain why this function won't do that, and show how to fix it. Your fix must be to the function only; you must not change the the main routine below in any way, yet as a result of your fixing the function, the main routine below must work correctly.

```
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. The hypotenuse function is correct, but the main function has a problem. Explain why it may not work, and show a way to fix it. Your fix must be to the main function only; you must not change the hypotenuse function in any way.

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

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

int main()
{
    double* p;
    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;

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()
{
    int junk[100];
    for (int k = 0; k < 100; k++)
        junk[k] = 123400000 + k;
}

int main()
{
    int m;
    int* ptr = computeSquares(m);
    f();
    for (int i = 0; i < m; i++)
        cout << ptr[i] << ' ';
    return 0;
}
```

2. For each of the following parts, write a single C++ statement that performs the indicated task. For each part, assume that all previous statements have been executed (e.g., when doing part e, assume the statements you wrote for parts a through d have been executed).

   a. Declare a pointer variable named fp that can point to a variable of type string.
   b. Declare fish to be a 5-element array of strings.
   c. Make the fp variable point to the last element of fish.
   d. Make the string pointed to by fp equal to "tilapia", using the * operator.
   e. Without using the fp pointer, and without using square brackets, set the fourth element (i.e., the one at index 3) of the fish array to have the value "salmon".

f. Move the fp pointer back by three strings.
g. Using square brackets, but without using the name fish, set the third element (i.e., the one at index 2) of the fish array to have the value "tuna".
h. Without using the * operator, but using square brackets, set the string pointed to by fp to have the value "ono".
i. Using the == operator in the initialization expression, declare a bool variable named d and initialize it with an expression that evaluates to true if fp points to the string at the start of the fish array, and to false otherwise.
j. Using the * operator in the initialization expression, but no square brackets, declare a bool variable named b and initialize it to true if the string pointed to by fp is equal to the string immediately following the string pointed to by fp, and false otherwise.

3.

a. Rewrite the following function so that it returns the same result, but does not increment the variable ptr. Your new program must not use any square brackets, but must use an integer variable to visit each double in the array. You may eliminate any unneeded variable.

```
double computeAverage(const double* scores, int nScores)
{
    const double* ptr = scores;
    double tot = 0;
    while (ptr != scores + nScores)
    {
        tot += *ptr;
        ptr++;
    }
    return tot/nScores;
}
```

b. Rewrite the following function so that it does not use any square brackets (not even in the parameter declarations) but does use the integer variable k. Do not use any of the <cstring> functions such as strlen, strcpy, etc.

```
// This function searches through str for the character chr.
// If the chr is found, it returns a pointer into str where
// the character was first found, otherwise nullptr (not
// found).

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.

4. What does the following program print and why? Be sure to explain why each line of output prints the way it does to get full credit.

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

int* minimart(int* a, int* b)
{
    if (*a < *b)
        return a;
    else
        return b;
}

void swap1(int* a, int *b)
{
    int* temp = a;
    a = b;
    b = temp;
}
```

```
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[1] = 9;
    ptr += 2;
    *ptr = -1;
    *(array+1) = 79;

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

    swap1(&array[0], &array[1]);
    swap2(array, &array[2]);

    for (int i = 0; i < 6; i++)
        cout << array[i] << endl;
    return 0;
}
```

5. Write a function named deleteG that accepts one character pointer as a parameter and returns no value. The parameter is a C string. This function must remove all of the upper and lower case 'g' letters from the string. The resulting string must be a valid C string.

   Your function must declare no more than one local variable in addition to the parameter; that additional variable must be of a pointer type. Your function must not use any square brackets and must not use the strlen or strcpy library functions.

```
    int main()
    {
        char msg[100] = "I recall the glass gate next to Gus in 
Lagos, near the gold bridge.";
        deleteG(msg);
        // prints I recall the lass ate next to us in Laos, near
        // the old bride.
        cout << msg;
        return 0;
    }
```

6. In this problem, you will write a program that can be used to gather statistical data about the number of movies college students see in a month. The program should perform the following steps:

   a. Ask the user how many students were surveyed. An array of integers with that many students should then be dynamically allocated.
   b. Allow the user to enter the number of movies each student saw into the array.
   c. Calculate and display the average, median, and mode of the values entered. Write functions for calculating the median and the mode.

   Input Validation: Do not accept negative numbers for input. If they attempt to put in a negative number, re-prompt until at least 0 is entered.