Project 6
What's the Point?

Time due: 9:00 PM Tuesday, November 22

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. There are five 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 << "10\n20\n30" << endl;
}
```

1. The subparts to this problem involve errors in the use of pointers.
   a. This program is supposed to write **10 20 30**, one per line. 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 << endl;    // print values
        }
    }
```

   r. The `findMax` function is supposed to find the maximum item in an array and set the `pToMax` parameter to point to that item so that the caller knows its location. 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 main routine below in any way, yet as a result of your fixing the function, the main routine below must work correctly.

```
s.      void findMax(int arr[], int n, int* pToMax)
t.      {
u.          if (n <= 0)
v.              return;      // no items, no maximum!
w.
x.          pToMax = arr;
y.
z.          for (int i = 1; i < n; i++)
aa.             {
bb.                 if (arr[i] > *pToMax)
cc.                      pToMax = arr + i;
dd.             }
ee.        }
ff.
gg.        int main()
hh.        {
ii.            int nums[4] = { 5, 3, 15, 6 };
jj.            int* ptr;
kk.
ll.            findMax(nums, 4, ptr);
mm.            cout << "The maximum is at address " << ptr << endl;
nn.            cout << "It's at index " << ptr - nums << endl;
oo.            cout << "Its value is " << *ptr << endl;
pp.        }
```

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

```
rr.        void computeCube(int n, int* ncubed)
ss.        {
tt.            *ncubed = n * n * n;
uu.        }
vv.
ww.        int main()
xx.        {
yy.            int* ptr;
zz.            computeCube(5, ptr);
aaa.           cout << "Five cubed is " << *ptr << endl;
bbb.       }
```

ccc. The `strequal` function is supposed to return true if and only if its two C string arguments have exactly same text. What are the problems with the implementation of the function, and how can they be fixed?

```
ddd.          // return true if two C strings are equal
eee.       bool strequal(const char str1[], const char str2[])
fff.       {
ggg.           while (str1 != 0  &&  str2 != 0)
hhh.           {
iii.               if (str1 != str2)  // compare corresponding
   characters
jjj.                   return false;
kkk.              str1++;            // advance to the next
   character
lll.              str2++;
mmm.          }
nnn.          return str1 == str2;   // both ended at same time?
ooo.      }
ppp.
qqq.      int main()
rrr.      {
sss.          char a[10] = "Bryan";
ttt.          char b[10] = "Bryant";
uuu.
vvv.          if (strequal(a,b))
www.              cout << "They're the same guy!\n";
xxx.      }
```

yyy. This program is supposed to write `5 4 3 2 1`, but it probably does not. What is the problem with this program? (We're not asking you to propose a fix to the problem.)

```
zzz.      int* getPtrToArray(int& m)
aaaa.     {
bbbb.         int anArray[5] = { 5, 4, 3, 2, 1 };
cccc.         m = 5;
dddd.         return anArray;
eeee.     }
ffff.
gggg.     void f()
hhhh.     {
iiii.         int junk[100];
jjjj.         for (int k = 0; k < 100; k++)
kkkk.             junk[k] = 123400000 + k;
llll.     }
mmmm.
nnnn.     int main()
oooo.     {
pppp.         int n;
qqqq.         int* ptr = getPtrToArray(n);
rrrr.         f();
ssss.         for (int i = 0; i < n; i++)
tttt.             cout << ptr[i] << ' ';
uuuu.         cout << endl;
vvvv.     }
```

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 `cat` that can point to a variable of type double.
   b. Declare `mouse` to be a 5-element array of doubles.
   c. Make the `cat` variable point to the last element of `mouse`.
   d. Make the double pointed to by `cat` equal to 17, using the * operator.

e. Without using the `cat` pointer, and without using square brackets, set the fourth element (i.e., the one at index 3) of the `mouse` array to have the value 42.
f. Move the `cat` pointer back by three doubles.
g. Using square brackets, but without using the name `mouse`, set the third element (i.e., the one at index 2) of the `mouse` array to have the value 33.
h. Without using the `*` operator, but using square brackets, set the double pointed to by `cat` to have the value 25.
i. Using the `*` operator in the initialization expression, declare a bool variable named `b` and initialize it to true if the double pointed to by `cat` is equal to the double immediately following the double pointed to by `cat`, and false otherwise.
j. Using the `==` operator in the initialization expression, declare a bool variable named `d` and initialize it to true if `cat` points to the double at the start of the `mouse` array, 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 computeMean(const double* scores, int numScores)
{
    const double* ptr = scores;
    double tot = 0;
    while (ptr != scores + numScores)
    {
        tot += *ptr;
        ptr++;
    }
    return tot/numScores;
}
```

m. 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`.

```
// 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 NULL (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 NULL;
}
```

z. 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* maxwell(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 = maxwell(array, &array[2]);
      *ptr = -1;
      ptr += 2;
      ptr[1] = 9;
      *(array+1) = 79;

      cout << &array[5] - ptr << endl;

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

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

48. Write a function named `removeS` 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 's' 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 `strcpy` library function.

```
int main()
{
    char msg[50] = "She'll be a massless princess.";
    removeS(msg);
    cout << msg;  // prints   he'll be a male prince.
}
```

Prepare your solutions to these homework problems as a single Word document named **hw.doc** or **hw.docx**, or a plain text file named **hw.txt**. Put that file in a zip file, and follow the turn in Project 6link to turn in the zip file.