CS31 **Summer 2014**

Assignment4
Who's Got the Point(er)?

1. Define integer variables x=2, y=12 and two pointer variables p and q (p pointer of **x**, and q pointer of **y)**. Then print the following information:

   a. The address of **x** and the value of **x**.
   b. The value of **p** and the value of ***p**.
   c. The address of **y** and the value of **y**.
   d. The value of **q** and the value of ***q**.
   e. The address of **p**.
   f. The address of **q**.

2. Define an array of integers **arr[5]** and fill this array with values 11, 40, 27, 0, 3. Define integer **i** and a pointer variable **p** that points to i. What is the output of following two printing loops:

```
for (i = 0; i < 5; i++)
   cout << i << " " << arr+i << " " << arr[i] << endl;
   cout << endl;

i = 0;
p = arr;

while (p < (arr+5))
   {
      cout << i << " " << p << " " << *p << endl;
      i++;
      p++;
   }
```

3. The subparts to this problem involve errors in the use of pointers.
   a. This program is supposed to write `50 60 70`, one per line. Find all of the bugs and show a fixed version of the program:

```
#include <iostream>
using namespace std;
    int main()
    {
        int MyArray[3] = { 10, 20, 30 };
        int *p = MyArray;

        *p = 70;          // set MyArray[0] to 70
        *p+= 1;
        *p= 60;      // set MyArray[1] to 60
        p+= 2;
        p[0] = 50;        // set MyArray[2] to 50

       do
       {
         p--;
         cout << *p << endl;    // print values
       }while (p >= MyArray);
    }
```

   b. The `findmin` function is supposed to find the minimum item in an array and set the `pToMin` parameter to point to that item so that the caller knows that item's 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.

```
void findmin(int MyArray[], int n, int* pToMin)
    {
        if (n <= 0)
            return;      // no items, no maximum!
        pToMin = MyArray;

        for (int i = 1; i < n; i++)
        {
            if (MyArray[i] < *pToMin)
                 pToMin = MyArray + i;
        }
    }

    int main()
    {
        int nums[4] = { 45, 13, 5, 66 };
        int* p;

        findmin(nums, 4, p);
        cout << "The minimum is at address " << p << endl;
        cout << "It's at position " << p - nums << endl;
        cout << "Its value is " << *p << endl;
    }
```

c. 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 anyway.

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

    void computeCube(int n, int* ncubed)
    {
        *ncubed = n * n * n;
    }

    int main()
    {
          int* ptr;
        computeCube(5, ptr);
        cout << "Five cubed is " << *ptr << endl;
    }
```

d. 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?

```
#include <iostream>
using namespace std;
// return true if two C strings are equal
bool strequal(const char str1[], const char str2[])
{
    while (str1 != 0  &&  str2 != 0)
    {
         if (str1 != str2)  // compare corresponding characters
         return false;
         str1++;            // advance to the next character
         str2++;
    }
        return *str1 == *str2;   // both ended at same time?
}
int main()
{
   char a[15] = "Chen, B.";
   char b[15] = "Chen, Y.J.";

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

e. 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.)

```
int* getPtrToArray(int& m)
{
    int anArray[5] = { 5, 4, 3, 2, 1 };
    m = 5;
    return anArray;
}

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

int main()
{
    int n;
    int* ptr = getPtrToArray(n);
    f();
    for (int i = 0; i < n; i++)
        cout << ptr[i] << ' ';
    cout << endl;
}
```

4. 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 `student` that can point to a variable of type double.
   b. Declare `grades` to be a 5-element array of doubles.
   c. Make the `student` variable point to the last element of `grades`.
   d. Make the double pointed to by `student` equal to 17, using the * operator.
   e. Without using the `student` pointer, and without using square brackets, set the fourth element (i.e., the one at position 3) of the `grades` array to have the value 72.
   f. Move the `student` pointer back by three doubles.
   g. Using square brackets, but without using the name `grades`, set the third element (i.e., the one at position 2) of the `grades` array to have the value93.
   h. Without using the `*` operator, but using square brackets, set the double pointed to by `student` to have the value 85.

5. 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 mean(const double* scores, int numScores)
{
    const double* ptr = scores;
    double tot = 0;
    while (ptr != scores + numScores)
    {
        tot += *ptr;
        ptr++;
    }
    return tot/numScores;
}
```

   a. 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;
}
```

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.

6. What does the following program print and why? Be sure to explain why each line of output prints the why 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;
}
```