Fall 2018 CS 31

**Project 6  
What's the Point?** (of grading this)

**Time due: 11:00 PM Monday, November 26**

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 << "30\n20\n10" << endl;

}

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

a. This program is supposed to write **30 20 10**, 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 = 30; // set arr[0] to 30

\*(ptr + 1) = 20; // set arr[1] to 20, there was a compile error on this line (left side of assignment did not resolve to a variable)

ptr += 2;

ptr[0] = 10; // set arr[2] to 10

ptr = arr;

while (ptr < arr + 3) // this loop printed them backwards

{

cout << \*ptr << endl; // print values

ptr++; // switched location of this statement

}

}

b. 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 that item's location. Explain why this function won't do that, and show a way 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.

The function does not change the pointer passed in since it passed in as a value. If we change it to passing by reference, the actual pointer instead of a copy of it will be modified to the intended value.

void findMax(int arr[], int n, int\*& pToMax) // changed to passing by reference

{

if (n <= 0)

return; // no items, no maximum!

pToMax = arr;

for (int i = 1; i < n; i++)

{

if (arr[i] > \*pToMax)

pToMax = arr + i;

}

}

int main()

{

int nums[4] = { 5, 3, 15, 6 };

int\* ptr;

findMax(nums, 4, ptr);

cout << "The maximum is at address " << ptr << endl;

cout << "It's at position " << ptr - nums << endl;

cout << "Its value is " << \*ptr << endl;

}

c. The computeCube 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 computeCube in any way.

The pointer is not initialized to any memory address. When the function tries to assign a value to pointer without an address, it throws an error. Fix: create integer variable and assign the address to pointer.

void computeCube(int n, int\* ncubed)

{

\*ncubed = n \* n \* n;

}

int main()

{

int x; // create variable x and its memory address

int\* ptr = &x; // set the pointer to that address

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. Explain what the problems with the implementation of the function are, and show a way to fix them.

The function uses the pointers instead of the values they point to (which was intended). This results in errors such as comparing a pointer to an integer.  
 // return true if two C strings are equal

bool strequal(const char str1[], const char str2[])

{

while (\*str1 != 0 && \*str2 != 0) // add stars

{

if (\*str1 != \*str2) // compare corresponding characters, add stars

return false;

str1++; // advance to the next character

str2++;

}

return \*str1 == \*str2; // both ended at same time?, add stars

}

int main()

{

char a[15] = "Zhou";

char b[15] = "Zou";

if (strequal(a,b))

cout << "They're the same person!\n";

}

e. This program is supposed to write 100 99 98 3 2 1, 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.)

The function getPtrToArray is returning a pointer that points to a locally declared array.

#include <iostream>

using namespace std;

int\* nochange(int\* p)

{

return p;

}

int\* getPtrToArray(int& m)

{

int anArray[100];

for (int j = 0; j < 100; j++)

anArray[j] = 100-j;

m = 100;

return nochange(anArray);

}

void f()

{

int junk[100];

for (int k = 0; k < 100; k++)

junk[k] = 123400000 + k;

junk[50]++;

}

int main()

{

int n;

int\* ptr = getPtrToArray(n);

f();

for (int i = 0; i < 3; i++)

cout << ptr[i] << ' ';

for (int i = n-3; i < n; i++)

cout << ptr[i] << ' ';

cout << endl;

}

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.

double\* cat;

b. Declare mouse to be a 5-element array of doubles.

double mouse[5];

c. Make the cat variable point to the last element of mouse.

cat = mouse + 4;

d. Make the double pointed to by cat equal to 25, using the \* operator.

\*cat = 25;

e. Without using the cat pointer, and without using square brackets, set the fourth element (i.e., the one at position 3) of the mouse array to have the value 42.

\*(mouse + 3) = 42;

f. Move the cat pointer back by three doubles.

cat -= 3;

g. Using square brackets, but without using the name mouse, set the third element (i.e., the one at position 2) of the mouse array to have the value 27.

cat[1] = 27;

h. Without using the \* operator or the name mouse, but using square brackets, set the double pointed to by cat to have the value 54.

cat[0] = 54;

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. Do not use the name mouse.

bool b = (\*cat == \*(cat + 1));

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.

bool d = (cat == mouse);

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 mean(const double\* scores, int numScores)

{

const double\* ptr = scores;

double tot = 0;

int i = 0;

while (i < numScores)

{

tot += \*(ptr + i);

i++;

}

return tot/numScores;

}

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)

{

const char\* ptr = str;

for (int k = 0; \*(ptr + k) != 0; k++)

if (\*(ptr + k) == chr)

return ptr + 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. Do not use any of the <cstring> functions such as strlen, strcpy, etc.

const char\* findTheChar(const char \* str, char chr)

{

for (const char\* ptr = str; \*ptr != 0; ptr++)

if (\*ptr == chr)

return ptr;

return nullptr;

}

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]); // since the value at the 0th position is greater than the 2nd position, ptr = array

\*ptr = -1; // it sets array[0] to -1 —> { -1, 3, 4, 17, 22, 19 }

ptr += 2; // ptr is now at the 2nd position

ptr[1] = 9; // sets 3rd position to 9 —> { -1, 3, 4, 9, 22, 19 }

\*(array+1) = 79; // sets 1st position to 79 —> { -1, 79, 4, 9, 22, 19 }

cout << &array[5] - ptr << endl; // outputs 3 since ptr is the pointer to the 2nd position so 5 - 2 = 3

swap1(&array[0], &array[1]); // the function manipulates copies of these pointers so they are not changed

swap2(array, &array[2]); // the values that these pointers point to are changed, so the values in the array are swapped —> { 4, 79, -1, 9, 22, 19 }

for (int i = 0; i < 6; i++)

cout << array[i] << endl; // outputs the array

}

Answer:

3

4

79

-1

9

22

19

5. 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 any of the <cstring> functions such as strlen, strcpy, etc.  
void removeS(char \* p) {

char \* s = p;

for (; \*p != ‘\0’; p++) {

if (\*p != ’s’ && \*p != ’S’) {

\*s = \*p;

s++;

}

}

\*s = 0;

}

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 6 link to turn in the zip file.