1.

a.

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

\*(ptr + 2) = 10; // set arr[2] to 10

for( ; ptr < arr+3 ; ptr++)

{

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

}

}

b.

The problem with the findmax function is the parameter pToMax is pass by value, even though it was set to the right address, but since the function only has a copy of pToMax, it will not be set correctly inside the main function. A simple fix to the problem is to add & and make it pass by reference.

void findMax(int arr[], int n, int\* &pToMax)

{

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 problem in the main function is because the pointer ptr

is declared without initializing, which leaves it pointing nothing

and the fix is :

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int x;

int\* ptr = &x;

computeCube(5, ptr);

cout << "Five cubed is " << \*ptr << endl;

}

d.

The intention for this function was to compare the value between the c-sting str1 and str2.

However, the function was missing the \* operator, and what the function was doing

was actually comparing the address of each element instead of the values.

//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++;

}

cout<<"gets here "<<endl;

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

}

int main()

{

char a[15] = "Zhou";

char b[15] = "Zou";

if (strequal(a,b))

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

else

cout << "They're t7he 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 problem with this program was related to variable accessing of the scope. The function int\* getPtrToArray(int& m)

tends to return a local variable, anArray. However, anArray was only recognized within the function getPtrToArray, once it was returned, it refers to a variable that no longer exist

which cause an undefined behavior in the main.

2.

a. double\* cat;

b. double mouse[5];

c. cat = &mouse[4];

d. \*cat = 25;

e. \*(mouse + 3) = 42;

f. cat -= 3;

g. cat[1] = 27;

h. cat[0] = 54;

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

j. bool d = (cat == &mouse[0])

3.

a.

double mean(const double\* scores, int numScores)

{

double tot = 0;

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

{

tot += \*(scores+i);

}

return tot/numScores;

}

b.

//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.

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

{

for ( ;\*str != 0; str++)

if ( \*str == chr)

return str;

return nullptr;

}

4.

#include <iostream>

using namespace std;

int\* maxwell(int\* a, int\* b)// compares the values of pass in addresses

{

if (\*a > \*b)

return a;

else

return b;

}

void swap1(int\* a, int\* b) // swap the address of the pass in addresses

{

int\* temp = a;

a = b;

b = temp;

}

// swap the values stored in the pass in addresses

void swap2(int\* a, int\* b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

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

//maxwell compares \*&array[0] and \*&array[2] and sets ptr to &array[0]

int\* ptr = maxwell(array, &array[2]);

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

ptr += 2; // moves the pointer to &array[2]

ptr[1] = 9; //sets element of &array[3] to 9 , { -1, 3, 4, 9, 22,19 }

\*(array+1) = 79; //sets array[1] to 79 { -1, 79, 4, 9, 22, 19 }

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

/\*ptr is now at &array[2],

this line finds the number entries between

&array[5] and &array[2], which prints out 3 \*/

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

//swap1 only swap the address between &array[0]and &array[1]but not the //values, so no effect on the order, {-1,79,4,9,22,19}

swap2(array, &array[2]);

//swap2 swaps the value at &array[0] and &array[2],

//result==> {4,79,-1,9,22,19}

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

cout << array[i] << endl;

}

outputs

**3**

**4**

**79**

**-1**

**9**

**22**

**19**

5.

void removeS( char\* cStr)

{

char\* ptr = cStr;

for(;\*cStr != 0 ; cStr++)

{

if( \*cStr != 'S' && \*cStr != 's')

{

\*ptr = \*cStr;

ptr++;

}

}

\*ptr = 0;

}

int main()

{

char msg[50] = "She'll be a massless princess.";

removeS(msg);

cout << msg; // prints he'll be a male prince.

}