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

ptr += 2;

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

while (ptr >= arr)

{

ptr--;

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

}

}

the bugs are underlined.

fixed version:

int main()

{

int arr[3] = { 5, 10, 15};

int\* ptr = arr;

\*ptr = 30;

\*(ptr + 1) = 20;

ptr +=2;

\*ptr = 10;

while (ptr <= arr+4)

{

cout << \*(ptr-2) << endl;

ptr++;

}

}

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.

void findMax(int arr[], int n, int\* pToMax)／／pass-by-value, wont change the ptr itself.

{

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;

}

the bugs are underlined

the fixed version:

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;

}

}

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.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int\* ptr; //the pointer ptr is not initialized!

computeCube(5, ptr);

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

}

the bugs are underlined.

the fixed version:

int main()

{

int\* ptr = new int; //set ptr to the address of an //unknown variable

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.

// return true if two C strings are equal

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

{

while (str1 != 0 && str2 != 0)//can’t write it //this way to say that the position has no character, this is just the position of the character in question, not the content.

{

if (str1 != str2) // compare corresponding characters //you can’t write it this way to express str1[0]and str2[0], str1 is equal to &str1[0] which is the address of str1[0]

return false;

str1++; // advance to the next character

str2++;//this is to move the address to the next character, but it is useless for the comparison above

}

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

} //same

int main()

{

char a[15] = "Zhao";

char b[15] = "Zhou";

if (strequal(a,b))

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

}

the bugs are underlined.

the fixed version:

// return true if two C strings are equal

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

{

char \* s1 = str1;

char \* s2 = str2;

while (\*s1 != ‘\0’ && \*s2 != ‘\0’)

{

if (\*s1 != \*s2) // compare corresponding characters

return false;

s1++; // advance to the next character

s2++;

}

return \*s1 == \*s2; // both ended at same time?

}

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.)

#include <iostream>

using namespace std;

int\* getPtrToArray(int& m)

{

int anArray[100];

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

anArray[j] = 100-j;

m = 100;

return anArray; //return the address of &anArray[0]

}

void f() //this function does nothing

{

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 < 3; i++)

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

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

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

cout << endl;

}

what it does wrong is declaring the array inside the getPtrToArray function, instead of declaring it in the main function and use it as parameter in the getPtrToArray function.

2.

a. double\* cat;

b. double mouse[5];

c. cat = mouse+4; //position 4

d. \*cat = 25;

e. \*(mouse-1) = 42;

f. cat -= 3; //position 1

g. \*(cat+1) =54;

h. cat[0] = 17; //???

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

j. bool d = (cat == mouse);

3.

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

{

const double\* ptr = scores;

double tot = 0;

int n = 0;

while (n < numScores)

{

tot += \*(ptr+n);

n++;

}

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\* s, char chr)//s is

//the address of the first entry in the array

{

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

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

return (s+k);

return nullptr;

}

c. // 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\* s, char chr)

{

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

if (\*s == chr)

return s;

return nullptr;

}

4.

int\* maxwell(int\* a, int\* b)

// this function returns the address of the larger value that pointers a and b point to.

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

//this function exchanges the address of pointer a and pointer b

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

//this function exchanges the values pointed by pointer a and pointer b

pseudocode:

set ptr equals to the address of array[0]//5>4

set array[0] equals 1

change the address of ptr to point array[2]

set the value of array[3] to 9

set array[1] to 79

output line1:3 //&array[5]-&array[2]

//pointer operation’s unit is always the

//position of things in the array

exchange the address pointed by &array[0] and &array[1]

exchange the value pointed by &array[0](which is the original address of &array[1]) and &array[2]

total output:

3

-1

4

79

9

22

19

5.

void removeS(char\* ors);

int main()

{

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

removeS(msg);

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

}

void removeS(char\* ors)

{

char\* nws = ors;

while(\*ors != 0)

{

if(tolower(\*ors) == ’s’)

{

ors++;

}

else

{

\*nws = \*ors;

ors++;

nws++;

}

}

\*nws = ‘\0’;

}