**1a.**

int main()

{

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

int\* ptr = arr;

\*ptr = 30;

\*ptr + 1 = 20; // need parentheses to increment the pointer

ptr += 2;

ptr[0] = 10;

while (ptr >= arr). //this is going backwards and skips one value,

// it will print 20,30,0

{

ptr--;

cout << \*ptr << endl;

}

}

Fixed Code:

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

int counter = ptr - arr;

while (counter >=0)

{

cout << \*(ptr - counter) << " ";

counter--;

}

}

**1b.**

//The function will not run correctly because the pointer is not passed by //reference

Fixed code:

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

{

if (n <= 0)

return; // no items, no maximum!

pToMax = arr;

for (int i = 0; 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;

}

**1c.**

//The function will not work because ptr has not been initialized

Fixed code:

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int temp = 0;

int\* ptr = &temp;

computeCube(5, ptr);

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

}

**1d.**

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

{

while (str1 != 0 && str2 != 0) //pointer needs to be dereferenced

{

if (str1 != str2) //pointer needs to be dereferenced

return false;

str1++;

str2++;

}

return str1 == str2; //here we should be comparing the values the

//pointers point to, not the pointers

//themselves

}

int main()

{

char a[15] = "Zhou";

char b[15] = "Zou";

if (strequal(a,b))

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

}

FIXED CODE:

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] = "Zhou";

char b[15] = "Zou";

if (strequal(a,b))

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

}

**1e.**

We are referencing the array which is inside of another function, so it is in the scope of that function. Once that function is done running, the array gets thrown away. If the array was a global variable array we would not have this problem. The program runs on undefined behavior.

**2.**

1. double\* cat;
2. double mouse[5];
3. cat = &mouse[4];
4. \*cat = 25;
5. \*(mouse + 3) = 42;
6. cat = cat – 3;
7. cat[1] = 27;
8. cat[0] = 54;
9. bool b = (\*cat == \*(cat+1));
10. bool d = (cat == &mouse[0]);

**3a.**

FIXEDCODE**:**

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

{

const double\* ptr = scores;

double tot = 0;

int x = 0;

while (x < numScores)

{

tot += \*(ptr+x);

x++;

}

return tot/numScores;

}

**3b.**

// 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).

FIXEDCODE**:**

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

{

for (int x = 0; \*(str + x) != ‘\0’; x++)

if (\*(str + x) == chr)

return (str + x);

return nullptr;

}

**3c.**

FIXEDCODE**:**

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

{

for (; \*str != ‘\0’; str++)

if (\*str == chr)

return str;

return nullptr;

}

**4.**

3 //ptr is at position 2, so the address of position 5 – the address of //position two is 3

4

79

-1

9

22

19

//First we print out three because ptr is at position 2, so the address of position 5 – the address of position two is 3. Next, we do a lot of array manipulation. So:

int\* ptr = maxwell(array, &array[2]); //ptr is 0;

\*ptr = -1; // array is now { -1, 3, 4, 17, 22, 19 }

ptr += 2; //ptr is now 2

ptr[1] = 9; // array is now { -1, 3, 4, 9, 22, 19 }

\*(array+1) = 79; // array is now { -1, 79, 4, 9, 22, 19 }

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

swap1(&array[0], &array[1]); // array has no change

//because swap1 only switches what the

//pointers point to, not the values.

swap2(array, &array[2]); //array is now { 4, 79,-1, 9, 22, 19 } as -1

//and 4 get swapped by swap2

for (int i = 0; i < 6; i++) // final array { 4, 79,-1, 9, 22, 19 }

cout << array[i] << endl;

**5.**

void removeS(char ch[])

{

char\* ptr = ch;

for (; \*ptr != ‘\0’; ptr++)

{

if (\*ptr == ‘s’ || \*ptr == ‘S’)

{

for (; \*ptr != ‘\0’; ptr++)

{

\*ptr = \*(ptr + 1);

}

ptr = ch;

}

}

}

int main()

{

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

removeS(msg);

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

}