**1 a.**

**Changes:**

added parenthesis: \*(ptr + 1)

moved ptr--; below the cout statement.

**Final Code:**

int main()

{

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

int\* ptr = arr;

\*ptr = 10; // set arr[0] to 10

\*(ptr + 1) = 20; // set arr[1] to 20

ptr += 2;

ptr[0] = 30; // set arr[2] to 30

while (ptr >= arr)

{

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

ptr--;

}

cout << endl;

}

**1 b.**

**Changes:**

Pass the pointer by reference.

**Final Code:**

void findDisorder(int arr[], int n, int\* &p)

{

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

{

if (arr[k] < arr[k - 1])

{

p = arr + k;

return;

}

}

p = nullptr;

}

int main()

{

int nums[6] = { 10, 20, 20, 40, 30, 50 };

int\* ptr;

findDisorder(nums, 6, ptr);

if (ptr == nullptr)

cout << "The array is ordered" << endl;

else

{

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

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

cout << "The item's value is " << \*ptr << endl;

}

}

**1 c.**

**Changes:**

Pointer p was uninitialized.

**Possible Fix 1:**

void hypotenuse(double leg1, double leg2, double\* resultPtr)

{

\*resultPtr = sqrt(leg1\*leg1 + leg2 \* leg2);

}

int main()

{

double x[] = {0};

double\* p = x;

hypotenuse(1.5, 2.0, p);

cout << "The hypotenuse is " << \*p << endl;

}

**Possible Fix 2:**

void hypotenuse(double leg1, double leg2, double\* resultPtr)

{

\*resultPtr = sqrt(leg1\*leg1 + leg2 \* leg2);

}

int main()

{

double x = 0;

double\* p = &x;

hypotenuse(1.5, 2.0, p);

cout << "The hypotenuse is " << \*p << endl;

}

**1 d.**

**Changes:**

Added '\*' to pointers to compare the objects of the pointers as well as changed 0 to the zero byte '\0'.

**Final Code:**

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

{

while (\*str1 != '\0' && \*str2 != '\0') // zero bytes at ends

{

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[10] = "Shankar";

char b[10] = "Shweta";

if (match(a, b))

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

}

**1 e.**

ptr depends on some unitialized array. This is undefined behavior.

**2.**

a. string\* fp;

b. string fish[5];

c. fp = (fish + 4);

d. \*fp = "yellowtail";

e. \*(fish+3) = "salmon";

f. fp = (fp - 3);

g. \*(fp + 1) = "loach";

h. p[0] = "eel";

i. bool d = (fp == &fish[0]);

j. bool b = (\*fp == \*(fp + 1));

**3 a.**

double computeAverage(const double\* scores, int nScores)

{

const double\* ptr = scores;

double tot = 0;

int k = 0;

while (k < nScores)

{

tot += \*(ptr + k);

k++;

}

return tot/nScores;

}

**3 b.**

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;

}

**3 c.**

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

{

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

if (\*curr\_char == chr)

return &curr\_char;

return nullptr;

}

**4.**

#include <iostream>

using namespace std;

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

{

if (\*a < \*b)

return a;

else

return b;

// 5 > 4 thus minimart returns b: pointer to the 3rd element of array

}

void swap1(int\* a, int \*b)

{

int\* temp = a;

a = b;

b = temp;

// creates pointers but doesn’t change array

}

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 = minimart(array, &array[2]);

// pointer created that points to the 3rd element of array

ptr[1] = 9;

// pointer is now 4, 9, 22, 19

// array is now 5, 3, 4, 9, 22, 19

ptr += 2;

// pointer now points to the 5th element of array

\*ptr = -1;

// 22 gets replaced with -1

// ptr is now -1, 19

// array is now 5, 3, 4, 9, -1, 19

\*(array+1) = 79;

// array becomes: 5, 79, 4, 9, -1, 19

cout << "diff=" << &array[5] - ptr << endl;

// The difference between the address of the sixth element of array and the address // of ptr[0] which points to the 5th element of array is 1 position which is why // this outputs 1.

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

// array is unchanged

swap2(array, &array[2]);

// array becomes: 4, 79, 5, 9, -1, 19

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

cout << array[i] << endl;

// for loop prints out the elements of the array: 4, 79, 5, 9, -1, 19

}

**5.**

void deleteG(char\* ptr)

{

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

{

if (\*(ptr + k) == 'g' || \*(ptr + k) == 'G')

{

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

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

}

}

return;

}