1a)

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, bugfix: needed parentheses

ptr += 2;

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

while (ptr >= arr)

{

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

ptr--; // bugfix: decrement needed after output

}

cout << endl;

}

1b)

The pointer variable was being passed by value to function, so the function only made changes to the copy of the variable that was made. For the pointer variable to be set properly, it needs to be passed by reference.

void findDisorder(int arr[], int n, int\*& p) //pointer passed by reference

{

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;

}

}

1c)

The pointer variable p is unitialized, so in attempting to assign a double to the memory location it points to results in undefined behavior in the program. p needs to be initialized to point to something.

#include <iostream>

#include <cmath>

using namespace std;

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

{

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

}

int main()

{

double d;

double\* p = &d; //p initialized to point to memory location of d

hypotenuse(1.5, 2.0, p);

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

}

1d)

The function compared the value of the pointers (the memory locations) rather than comparing what the pointers were pointing to.

// return true if two C strings are equal

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] = "Zhang";

char b[10] = "Zhao";

if (match(a,b))

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

}

1e)

Because arr is local to the function, the function ends up returning the address of a local variable. This makes it so that upon leaving the function, this pointer points to undefined memory, and attempting to access it results in undefined behavior.

-----------------------------------------------------------------------------

2)

using namespace std;

int main()

{

string\* fp; //a

string fish[5]; //b

fp = &fish[4]; //c

\*fp = "yellowtail"; //d

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

fp -= 3; //f

fp[1] = "perch"; //g

fp[0] = "eel"; //h

bool d = (fp == fish) //i

bool b = (\*fp == \*(fp+1)) //j

}

-----------------------------------------------------------------------------

3a)

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

{

const double\* ptr = scores;

double tot = 0;

int k = 0;

while (ptr + k != scores + nScores)

{

tot += \*(ptr + k);

k++;

}

return tot/nScores;

}

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).

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;

}

3c)

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

{

while(\*str != 0)

{

if (\*str == chr)

return str;

str++;

}

return nullptr;

}

-----------------------------------------------------------------------------

4)

-6 member array of ints declared: { 5, 3, 4, 17, 22, 19 }

-Pointers to position 0 and 2 are passed to function minimart

-Minimart function finds which pointer points to the smaller number and returns its location

-Pointer ptr is set to point to position 2 of the array

-The memory location 1 after ptr (position 3 of the array) is set to 9

{ 5, 3, 4, 9, 22, 19 }

-ptr is incremented to point two memory locations forward (position 4 of the array)

-what ptr points to is set to -1 { 5, 3, 4, 9, -1, 19 }

-The second element of the array is set to 79 { 5, 79, 4, 9, -1, 19 }

-The difference between the addresses of the 6th element of the array (position 5) and of ptr (position 4) is output: "diff=1"

-The addresses of the elements in positions 0 and 1 are passed to function swap1

-swap1 only affects local variables, as its parameters are not passed by reference

-The addresses of the elements in positions 0 and 2 are passed to function swap2

-swap2 swaps the values each pointer points to (elements in positions 0 and 2 are swapped) { 4, 79, 5, 9, -1, 19 }

-Elements of the array are printed out in order vertically

Output:

diff=1

4

79

5

9

-1

19

-----------------------------------------------------------------------------

5)

void deleteG(char\* str)

{

while(\*str != 0)

{

if(\*str == 'G' || \*str == 'g')

{

char\* s = str;

while(\*str != 0)

{

\*str = \*(str + 1); //everything after a ‘g’ moved left

str++;

}

str = s - 1;

}

str++;

}

}