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

//Wrong as it needs parentheses to indicate that the object at ptr+1 is set //to 20

ptr += 2;

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

while (ptr >= arr)

{

ptr--;

cout << ' ' << \*ptr; // print values//Needs to switch order

}

cout << endl;

}

Fixed:

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;

}

1b)

The pointer value is passed by value into the function. Therefore, even it finds the right pointer value, it doesn’t modify the pointer value outside the findDisorder function. For the ptr to be modified the function should have called for a reference to pointer.

Fixed:

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

1c)

The pointer double\* p is not pointing to anything. So, when it tries to retrieve the object it points to, in the cout line, it can’t retrieve anything. Therefore, to fix this we should first initialize a double variable and store its address to double \*p.

Fixed:

int main()

{

double hypo;

double\* p = &hypo;

hypotenuse(1.5, 2.0, p);

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

}

1d)

The problem with the code in the spec is that str1 and str2 are not dereferenced when a comparison is being made. They are comparing the memory address of str1 and str2 and not comparing the object that str1 and str2 pointers are pointing to.

Fixed:

*// 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?*

}

1e)

int\* computeSquares(int& n) returns a memory associated with local variable 'arr'. Outside of the computeSquares function, the returned memory address does not exist.

2a)

string \*fp;

2b)

string fish[5];

2c)

fp = &fish[4];

2d)

\*fp = "yellowtail";

2e)

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

2f)

fp = fp – 3;

2g)

fp[1] = "carp";

2h)

fp[0] = "smelt";

2i)

bool d = (fp == fish);

2j)

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

3a)

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

{

double tot = 0;

for (int i = 0; i < nScores; i++) {

tot += \*(scores + i);

}

return tot/nScores;

}

3b)

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)

{

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

if (\*str == chr) {

return str;

}

}

return nullptr;

}

4)

#include <iostream>

using namespace std;

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

*//returns pointer value to whichever object that a or b points to is smaller*

*//If they are equal the function returns b*

{

if (\*a < \*b)

return a;

else

return b;

}

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

*//Swaps local pointers a and b*

*//Doesn't swap anything outside of function*

{

int\* temp = a;

a = b;

b = temp;

}

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

*//Correctly swaps the int where a is pointing to where b is pointing to*

*//and vice versa*

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

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

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

*//Declares and initialize ptr as pointer to 4 (ie. &array[2]) as 4 < 5 (this is what //the minimart function does)*

ptr[1] = 9; *//Sets array[3] (ie 17) becomes 9*

ptr += 2; *//ptr points to &array[4] (ie 22)*

\*ptr = -1; *//Sets array[4] to -1 (ie 22 becomes -1)*

\*(array+1) = 79; *//Sets array[1] to 79 (ie 3 becomes 79)*

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

*//prints out diff = 1 (as &array[5] points to 5th index*

*//and ptr is pointing to 4th index therefore*

*//diff is 5th index - 4th index = 1*

swap1(&array[0], &array[1]); *//Doesn't do anything to the array*

swap2(array, &array[2]); *//Object at array[0] and array[2] switch places*

*//(ie 5 and 4 switches places)*

*//array is now {4, 79, 5, 9, -1, 19}*

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

cout << array[i] << endl;

*//Prints out all elements of the array*

}

Output:

diff=1

4

79

5

9

-1

19

5)

void deleteG(char \*str) {

char \*deleted = str;

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

if (\*str != 'G' && \*str != 'g') {

\*deleted = \*str;

deleted++;

}

}

\*deleted = '\0';

}