1.a.

Bugs:

* \*ptr + 1 need a bracket
* the order of output is reversed

fixed version:

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

while (ptr <= arr+2)

{

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

ptr++;

}

}

1.b.

Reason:

The pToMax parameter is not a reference type, so the value of ptr in the main routine won’t change.

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;

}

}

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;

}

1.c.

Problem: ptr is not initialized;

fixed:

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int a;

int\* ptr = &a;

computeCube(5, ptr);

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

}

1.d.

Problem:

It compares the pointers directly, which are addresses, but not the contents in the addresses.

fixed:

// return true if two C strings are equal

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] = "Tang";

char b[15] = "Zhang";

if (strequal(a,b))

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

}

1.e.

Because anArray is only defined in the function getPrtToArray, the function returns an address of memory that will be released after implementation of the function. So ptr may point to some junk value.

2.

int main()

{

double\* cat;

double mouse[5];

cat = mouse + 4;

\*cat = 42;

\*(mouse+3) = 17;

cat -= 3;

cat[1] = 25;

cat[0] = 54;

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

bool d = (cat == mouse);

}

3.a.

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

{

double tot = 0;

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

tot += \*(scores + i);

return tot/numScores;

}

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

if (\*str == chr)

return str;

return nullptr;

}

4.

int main()

{

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

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

\*ptr = -1; // array[0] = -1

ptr += 2; // ptr = array + 2

ptr[1] = 9; // array[3] = 9

\*(array+1) = 79; // array[1] = 79;

// array = { -1, 79, 4, 9, 22, 19 }

cout << &array[5] - ptr << endl; // array[5]=array+5, ptr = array + 2, the output is 3

swap1(&array[0], &array[1]); // nothing is changed

swap2(array, &array[2]); // array[0] and array[2]’s values are exchanged

// array = { 4, 79, -1, 9, 22, 19}

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

cout << array[i] << endl;

}

3

4

79

-1

9

22

19

5.

void removeS(char\* ptr)

{

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

if (\*ptr == 's' || \*ptr == 'S')

{

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

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

ptr--;

}

}