Question 1a

The first bug is in the line: \*ptr + 1 = 20; // set arr[1] to 20

The second bug is in the line: while (ptr >= arr)

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;

ptr[0] = 10; // set arr[2] to 10

while (ptr >= arr)

{

ptr--;

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

}

}

Question 1b

The parameter pToMax in function finMax is a copy of ptr in the main routine. Changing the value of pToMax does not modify ptr in the main routine. After execution of the function ptr is still pointing to num[0]. So, the program wouldn’t work.

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;

}

}

Question 1c

The ptr in the main function is not initialized. Using uninitialized variables may cause undefined behaviors. The pointer is pointing to random value.

Fixed version:

int main()

{

int a;

int\* ptr = &a;

computeCube(5, ptr);

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

}

Question 1d

The while statement is comparing 2 pointers with 0, but they are addresses. In the while loop, str1 and str2 are all pointers, and they are not equal. So, the function returns false. This function is comparing the pointer, not the object the pointer points to.

Fixed version:

// return true if two C strings are equal

bool strequal(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[15] = "Wang, A.";

char b[15] = "Wang, R.";

if (strequal(a,b))

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

}

Question 1e

After execution of function getPtrToArray, the int array anArray is recycled in memory, but the pointer in the main function still exists. The pointer is still pointing to the int object in the array. So, the pointer is pointing to some trash. This may produce unexpected results.

Question 2

1. double\* cat;
2. double mouse[5];
3. cat = mouse + 4;
4. \*cat = 25;
5. \*(mouse + 3) = 54;
6. cat -= 3;
7. cat[1] = 17;
8. cat[0] = 42;
9. bool d = (cat == mouse);
10. bool b = (\*cat == (\*cat + 1));

Question 3a

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

{

const double\* ptr = scores;

double tot = 0;

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

tot += \*(ptr + i);

}

return tot/numScores;

}

Question 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;

}

Question 3c

// 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)

{

while (\*str != 0){

if (\*str == chr) {

return str;

}

str++;

}

return nullptr;

}

Question 4

Output is:

3

4

79

-1

9

22

19

int main()

{

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

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

// setting ptr to the pointer that has the maximum value that it points to.

// 5 > 4

// in this case ptr is pointing to array[0]

\*ptr = -1;

// setting the value the pointer is pointing to to -1

// array[6] = { -1, 3, 4, 17, 22, 19 };

ptr += 2;

// pointing the pointer to the third object of array

// in this case ptr is pointing to array[2]

ptr[1] = 9;

// setting array[3] = 9;

// array[6] = { -1, 3, 4, 9, 22, 19 };

// ptr is still pointing array[2]

\*(array+1) = 79;

// setting array[1] = 79;

// array[6] = { -1, 79, 4, 9, 22, 19 };

// ptr is still pointing array[2]

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

// &array[5] is the pointer (address) to the last (6th) object in the array.

// ptr points to the 3rd object in the array.

// 5 - 2 = 3

// outputs:

//3

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

// swap1 swaps the input pointers

// that's a copy of pointers, and doesn't modify value of the pointer

swap2(array, &array[2]);

// swap2 swaps the value that the input pointers point to

// array[6] = { 4, 79, -1, 9, 22, 19 };

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

cout << array[i] << endl;

// outputs:

//4

//79

//-1

//9

//22

//19

}

Question 5

void removeS(char\* str) {

char\* temp = str;

while (\*temp != 0) {

if (\*temp == 's' || \*temp == 'S'){

str = temp;

while (\*str != 0) {

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

str++;

}

temp--;

}

temp++;

}

}