Ryan Lam

1.

a.

Since the function decrements ptr backwards when printing the array values, the array must be set to 10 20 30 instead of 30 20 10. You must also decrement ptr after printing the array value. You also need parentheses enclosing ptr + 1, so you are referring to the value at \*(ptr + 1) instead of trying to add 1 to \*ptr.

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 << endl; // print values

ptr--;

}

}

b.

FindMax is incorrect because it passes a local instance of pointer ptr into the function, rather than the actual pointer itself. Ptr is uninitialized in the main function, so the local instance of ptr would point to nothing and would not affect ptr in main. To fix this, you can use the ampersand symbol in the function parameter to pass a pointer to ptr, allowing the function to access and initialize ptr in findMax.

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;

}

c.

This function will not work because pointer ptr is declared in the main function but not initialized. You can fix this by initializing the pointer to some value, such as a new int.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int\* ptr = new int;

computeCube(5, ptr);

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

}

d.

The problem is that the function compares the pointers themselves instead of the values of those pointers. Comparing pointers tests if they point to the same address, not if their values are the same. For the function to work correctly, you must use the asterisk symbol to reference the value that the pointer points to in order to compare characters.

// 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] = "Zhou";

char b[15] = "Zou";

if (strequal(a,b))

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

}

e.

The program is declaring and initializing arrays within the functions as local variables but trying to access their values through pointers in the main function, causing the program logic to be incorrect.

2.

a. double\* cat;

b. double mouse[5];

c. cat = &mouse[4];

d. \*cat = 25;

e. \*(mouse + 3) = 42;

f. cat -= 3;

g. cat[1] = 27;

h. cat[0] = 54;

i. bool b = (\*cat == cat[1]);

j. bool d = (cat == mouse);

3.

a.

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

{

const double\* ptr = scores;

double tot = 0;

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

{

ptr = scores + i;

tot += \*ptr;

}

return tot/numScores;

}

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;

}

c.

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

{

while(str != 0)

{  
 if(\*str == chr)

return str;

str++;

}

return nullptr;

}

4.

The program prints:

3 : The 7th line subtracts a pointer from another pointer, resulting in the numerical difference between the two pointer addresses divided by the size of the pointer objects, or the separation between values in the array.

4 : The swap2 function swapped the values of index 0 and 2 in the array, which were -1 and 4 respectively before the swap.

79 : The 6th line references the value of the pointer to index 0 of the array plus 1, since array is equivalent to &array[0], therefore referencing the value of the pointer to index 1 of the array, changing it to 79.

-1 : In line 3, ptr initially pointed to index 0 of the array and changed its value to be -1. The swap1 function attempted to swap the pointers to index 0 and 1 to be pointing at each other’s indexes, but because the pointer parameters to swap1 are not the actual pointers to the indexes, the function does not do anything.

9 : Ptr initially pointed to index 0 of the array. In line 4, ptr was incremented by 2 and then, in line 5, referenced the object after the pointer using square brackets, ultimately pointing to the 4th value in the array and changing it to 9.

22 : Unchanged by program

19 : Unchanged by program

5.

void removeS(char\* ptr)

{  
 while(\*ptr)

{  
 if(\*ptr == ‘S’ or \*ptr == ‘s’)

{

char\* temp = ptr;

while(\*temp)

{  
 \*temp = \*(temp+1);

temp++;

}

ptr--;

}

ptr++;

}

}