Preethika Kiruveedula

204912895

Computer Science 31

Project 6

1)

a.

int main()

{

int arr[3] = { 5, 10, 15 };

int\* ptr = arr;

\*ptr = 10;

\*(ptr + 1) = 20;

\*(ptr + 2) = 30;

ptr += 2;

while (ptr >= arr)

{

cout << \*ptr << endl;

ptr--;

}

}

b.

You need to pass pToMax by reference in order for its initialization in the findMax fuction to transfer to the \*ptr in the main function. I made the third parameter in findMax pass by reference instead of pass by value

void findMax(int arr[], int n, int\*& pToMax)

{

if (n <= 0)

return;

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.

The reason this program doesn't work is because computeCube is a void function; therefore, it has no return value. Since it has no return value you cannot initialize the value to ptr and simply return it. You have to put in an ampersand so it becomes a reference parameter, and all the changes made to the ptr variable in the void function are transferred over to the main. I created a new integer variable and stored the value from computeCube into it and made it equivalent to \*ptr.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int i = 0;

int\* ptr = &i;

computeCube(5, ptr);

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

}

d.

The problem is that it is that it was comparing the memory addresses instead of the content so it would always return false because the memory addresses will never be equal. That’s why I changed str1 and str2 to const char\*and compared the pointers instead of the memory addresses.

bool strequal(const char\* str1, const char\* str2)

{

while (\*str1 == \*str2)

{

if (\*str1 == '\0' || \*str2 == '\0')

break;

str1++;

str2++;

}

if (\*str1 == '\0' && \*str2 == '\0')

return true;

else

return false;

}

int main()

{

char a[15] = "Zhou";

char b[15] = "Zhu";

if (strequal(a, b))

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

return 0;

}

e.

What this function implementation is doing incorrectly is that it is returning anArray as the value for the pointer in the main function. anArray is a local variable and it is not in the scope of the main function; therefore, the array is uninitialized and cannot be assigned to the ptr.

2)

a. double\* cat;

b. double mouse[5];

c. cat = mouse + 4;

d. \*cat = 25;

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

f. cat = cat - 3;

g. cat[1] = 54;

h. cat[0] = 27;

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

j. bool d = (\*cat == mouse[0]);

3)

a.

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

{

const double\* ptr = scores;

double tot = 0;

int i = 0;

while (ptr != scores + numScores)

{

tot += \*(ptr + i);

i++;

}

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\* findTheCharRewrite2(const char\* str, char chr)

{

while (\*str != '\0')

{

if (\*str == chr)

return str;

str++;

}

return nullptr;

}

4)

The output is:

3

4

79

-1

9

22

19

The first line of output is three because it is the difference between &array[5] (which is equal to 5) and ptr (which is equal to 2), 5 - 2 = 3

The rest of the lines of ouput are the elements of the array starting at the zero index and ending at the fifth index:

The second line of ouput is 4, the zero index of the array. The zero index is 4 because it was first initialized to \*ptr. The \*ptr value was then initialized to -1 and the zero index also changed to -1. Then it goes through the first swap and its value remains to be -1. Then once it goes through the second swap its memory address is swapped with that of array[2] (which was equal to 4) causing array[0] = 4 and array[2] to equal -1

The third line of output is 79, the first index of the array. array[1] is 79 because it was first initialized to be 3, then the function changed it to 79 by moving the pointer over one element and initializing it to 79. This element then goes through the first swap, but its value remains to be 79; therefore array[1] is equal to 79

The fourth line of output is -1, the second index of the array. It was initially equal to 4; however, after going through swap2 it switched memory addresses with the zero index of the array which used to equal -1 and array[2] = -1 and array[0] = 4

The fifth line of output is 9 because when the pointer was moved to point at array[3] the value next to it was initialized to be 9 by ptr + 1 = 9, meaning one element over from the third index element is now equal to 9.

The sixth line of output is 22 because it was not affected by the implementation of the function and remained the same as its initial value.

The seventh and final line of output is 19 because it was not affected by the implementation of the function and remained the same as it initial value.

5)

void removeS(char\* letter)

{

while (\*letter != '\0')

{

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

{

char\* c = letter + 1;

while (\*c != '\0')

{

\*(c - 1) = \*c;

c++;

}

if (\*c == '\0')

{

\*(c - 1) = \*c;

}

letter--;

}

letter++;

}

}

int main()

{

char s[50] = "She'll be a massless princess.";

removeS(s);

cout << s;

return 0;

}