1a))

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 // star operator has higher precedence than addition

ptr += 2;

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

int i = 0;

while (ptr >= arr + i) // this is counting backwards, need to start at beginning of array and count up until where the pointer is( at the last spot in the array)

{

~~ptr--;~~

cout << ~~\*ptr~~ \*(arr + i) << endl; // print values

i++;

}

}

1b))

void findMax(int arr[], int n, int\*& pToMax) pointer is being passed as a copy so the function will not change value of pointer

{

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 = &nums[0];

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;

}

1c))

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int x; //pointer is uninitialized, not pointing at anything

int\* ptr = &x;

computeCube(5, ptr);

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

}

1d))

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] = "Chang";

char b[15] = "Zhang";

if (strequal(a,b))

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

}

Explanation:

Instead of passing str1 and str2 as arrays, if they are passed as pointers they can be iterated to look inside of the array. Additionally, when comparing the corresponding characters or null character, we need to be looking at where the pointers are pointing.

1e))

You can’t declare an array in a function because that memory is no longer allocated to storing the array when the function has finished executing; the array lives within the scope of the function, so its values cannot be printed later on.

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] = 42;
8. cat[0] = 17;
9. bool d = (cat == mouse);
10. bool b = (\*cat == \*(cat + 1));

3)

a)

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

{

const double\* ptr = scores;

double tot = 0;

int i = 0;

while ((ptr + i) != 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\* findTheChar(const char \* str, char chr)

{

while(\*str != 0){

if(\*str == chr)

return str;

str++;

}

return nullptr;

}

4)

maxwell returns a pointer to the larger number, or if they are equal, to the second input.

swap1 does nothing.

swap2 swaps values at given locations.

ptr is set to beginning of array. first value is set to -1. ptr is moved to third value of array. one value past that (fourth value of array) is set to 9. second value of array is set to 79.

First output: Finds how far that end of the array is from the pointer.

first value and third value of array are swapped.

Second Output: outputs every value in the array.

Output:

3

4

79

-1

9

22

19

5)

void removeS (char\* ptr){

char\* ptr2 = ptr;

while (\*ptr != 0){

if(\*ptr == ‘s’ || \*ptr == ‘S’)

ptr++;

else{

\*ptr2 = \*ptr;

ptr++;

ptr2++;

}

}

\*ptr2 = 0;

}