1a. int main()

{

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

int\* ptr = arr;

\*ptr = 30;

ptr += 1;

\*ptr = 20;

ptr += 1;

\*ptr = 10;

ptr = arr;

int i = 0;

while (i < 3)

{

cout << \*ptr << endl;

ptr++;

i++;

}

}

1b. ptr is not passed by reference,so its value doesn’t change, and in the main function, it remains uninitialized.

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;

}

1c. ptr must be initialized before being passed to computeCube, because computeCube is trying to change the value of the int ptr points to, but ptr does not point to an int.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int x = 5;

int\* ptr = &x;

computeCube(x, ptr);

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

}

1d. It needs to reference each individual character, which can be done with a pointer. By creating a pointer, you can compare if every character of the cstring is equal, returning false if it is not.

bool strequal(char str1[], char str2[])

{

char\* c1 = str1;

char\* c2 = str2;

while (\*c1 != '\0' && c2 != '\0')

{

if (\*c1 != \*c2)

return false;

c1++;

c2++;

}

return (\*c1 == '\0' && \*c2 == '\0');

}

int main()

{

char a[15] = "Chen";

char b[15] = "Cheng";

if (strequal(a, b))

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

}

1e. ptr points to the first element of anArray, which doesn’t exist outside of getPtrToArray. Therefore, anArray ceases to exist, so it results in undefined behavior.

2a. double\* cat;

2b. double mouse[5];

2c. cat = mouse + 4;

2d. \*cat = 25;

2e. \*(mouse + 3) = 54;

2f. cat -= 3;

2g. cat[1] = 27;

2h. cat[0] = 42;

2i. bool b = (\*cat == \*(cat + 1));

2j. bool d = (\*cat == mouse[0]);

3a.

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

{

double\* ptr = scores;

double tot = 0;

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

{

double cur = \*(ptr + 1);

tot += cur;

}

return tot/numScores;

}

3b.

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

{

for (int k = 0; \*(ptr + k)!= ‘\0’; k++)

{

if (\*(ptr + k) == chr)

return ptr;

}

return nullptr;

}

3c.

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

{

while (\*ptr != ‘\0’)

{

if (\*ptr == chr)

return ptr;

ptr++;

}

return nullptr;

}

4. The program prints:

3

4

79

-1

9

22

19

ptr points to the first element of the array, 5. It is then decremented to 4. ptr is changed to point to the third element, 4. The fourth element is changed to 9. The second element is changed to 79. The first cout line prints the distance between the position that ptr points to and the position of the last element, which is three. swap1 is called, which changes which element each pointer points to, but doesn’t change the value of the array at that point. swap2 is called, and it swaps the values of the first and third element of the array. It then prints every element of the array.

5.

void removeS(char\* ptr)

{

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

{

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

{

for (int j = 0; \*(ptr + j) != '\0'; j++)

{

if (\*(ptr + i + j + 1) == '\0')

{

\*(ptr + i + j) = '\0';

}

else

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

}

i--;

}

}

}