1. a.

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

ptr = arr;

while (ptr <= (arr + 2))

{

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

ptr++;

}

}

1. b.

The function won't do what is intended because the parameter pToMax needs to be passed by reference so that it changes the value of pToMax outside of the function. This can be achieved by adding the & to the pointer parameter 🡺 int\*& pToMax

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;

}

}

1. c.

The main function will not work because the pointer was never initialized to a value, so it would result in an error.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int arr[1];

int\* ptr = arr;

computeCube(5, ptr);

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

}

1. d.

The problem with the function is that we want to compare the elements of the two C-strings. The way that the function is set up now, it is comparing the references of the C-strings, but if we want to compare the elements of the C-strings, then we need to use pointers to point to the elements.

In addition, the function is increasing the C-strings with the increment operator, but this operation cannot be done, and would result in undefined behavior if these lines were executed. To iterate through the C-strings, and with the use of pointers, the position that the pointers are pointing to will be incremented to stem through the array.

// return true if two C strings are equal

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

{

const char\* ptr1 = str1;

const char\* ptr2 = str2;

while (\*ptr1 != '\0' && \*ptr2 != '\0')

{

if (\*ptr1 != \*ptr2) // compare corresponding characters

return false;

ptr1++; // advance to the next character

ptr2++;

}

return \*ptr1 == \*ptr2; // both ended at same time?

}

1. e.

The problem with the program is that the array of the function getPtrToArray is lost when it is called. The array, anArray, is a local array and it is lost when returning to the main function. Another problem occurs from the function f, which assigns values that are out of range for the int type. This results in an overflow of data that results in undefined behavior.

2.

a. double\* cat;

b. double mouse[5];

c. cat = mouse + 4;

d. \*cat = 42;

e. \*(mouse + 3) = 25;

f. cat -= 3;

g. cat[1] = 17;

h. cat[0] = 54;

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

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

3. a.

double mean(const double\* scores, int numScores) //scores = 5, numScores = 10

{

const double\* ptr = scores;

double tot = 0;

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

{

tot += \*(ptr + i);

}

return tot/numScores;

}

3. b.

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

{

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

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

return (ptr + k);

return nullptr;

}

3. c.

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

{

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

if (\*ptr == chr)

return ptr;

return nullptr;

}

4.

//This function will return a pointer

int\* maxwell(int\* a, int\* b)

{

if (\*a > \*b)

return a;

else

return b;

}

//This function will swap the address of two numbers (pass by value)

void swap1(int\* a, int\* b)

{

int\* temp = a;

a = b;

b = temp;

}

//This function will swap the values of two numbers (pass by reference)

void swap2(int\* a, int\* b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

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

int\* ptr = maxwell(array, &array[2]); //Returns &array[0] 🡺 ptr = \*array[0]

\*ptr = -1; //sets array[0] = -1

ptr += 2; //Moves pointer to point to &a[2]

ptr[1] = 9; //sets array[3] = 9

\*(array+1) = 79; //sets array[1] = 79

cout << &array[5] - ptr << endl; //Prints out 3

//because &array[5] is position 6 and ptr is pointing to position 3 🡺 6 – 3 = 3

swap1(&array[0], &array[1]); //Swaps the positions of &a[0] and &a[1]; however, the function

//was pass by value and thus nothing really happened when it returned back to main function

swap2(array, &array[2]); //Swaps the values of the elements of array at array[0] and array[2]

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

cout << array[i] << endl;

//Prints out the whole array vertically 🡺 {4, 79, -1, 9, 22, 19}

Output:

3

4

79

-1

9

22

19

}

5.

void removeS(char\* msg)

{

char\* newMsg = msg;

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

{

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

{

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

{

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

newMsg++;

}

newMsg = msg;

}

newMsg++;

}

}