1.

1. The first thing is making \*ptr + 1 into \*(ptr+1) since the point of the code is to modify the element 1 in the arr. The next thing is to have a line saying ptr -= 2 so that the address is reset to be at the element 0. The while loop is changed so it prints in the correct order rather than the reverse order. Within the while loop, the cout needs to go before the ptr incrementation so it prints the indexes 0, 1, and 2.

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 -= 2;

while (ptr < &arr[3])

{

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

ptr++;

}

}

1. The problem with the code is that when the pointer is passed into the function findMax, a copy of that pointer is created. Therefore the value of the pointer is not changed within the main method. In order to fix this, the pointer should be passed by reference.

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. The problem with the previous code was that the pointer ptr was not initialized to any value before. Since the ptr is passed by reference, it must have an initialized address before it can hold the value of ncubed. The solution is to create a temporary integer and set pointer to the address of that integer. Then the method can be called and value of pointer can be properly changed.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int hold = 0;

int\* ptr = &hold;

computeCube(5, ptr);

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

}

1. There are two main problems with implementation. The first problem is that within the while loop the str1 and str2 should be dereferenced so it can be compared to the 0 byte. The second problem is that within the if statement the str1 and str2 should be dereferenced so its comparing the values at the address rather than the address itself.

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?

}

1. The problem with the code is that the array is defined in function getPtrToArray, and therefore restricted to the scope of that function. The compiler deletes the memory of the array created once the method is called. There is no array to access when trying to initialize the pointer. Therefore ptr cannot be initialized to the getPtrToArray method call.

2.

1. double\* cat;
2. double mouse[5];
3. cat = &mouse[4];
4. \*cat = 25;
5. \*(mouse + 3) = 42;
6. cat -= 3;
7. cat[1] = 27;
8. cat[0] = 54;
9. bool b = (\*cat == \*(cat+1));
10. bool d = (cat == &mouse[0]);

3.

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

{

double tot = 0;

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

{

tot += \*(scores + i);

}

return tot/numScores;

}

1. 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;

}

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

{

while (\*(str) != 0)

if (\*str == chr){

return str;

}

else{

str++;

}

return nullptr;

}

4. int\* maxwell(int\* a, int\* b) **//returns the pointer that has the greater value when dereferenced**

{

if (\*a > \*b)

return a;

else

return b;

}

void swap1(int\* a, int\* b) **//swaps the addresses between two pointers**

{

int\* temp = a;

a = b;

b = temp;

}

void swap2(int\* a, int\* b) **//swaps the value between two pointers**

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

int array[6] = { 5, 3, 4, 17, 22, 19 }; **//initializes an array of size 6 elements**

int\* ptr = maxwell(array, &array[2]); **//initializes a pointer variable equal to the value of the maxwell call. maxwell takes the address of elements 0 and 2, determines that 5 > 4. the pointer is set equal to 0.**

\*ptr = -1; **//array[0] is now equal to -1**

ptr += 2; **//increments the address to 2**

ptr[1] = 9; **//array[3] is now set equal to 9**

\*(array+1) = 79; **//array[1] is now set equal to 79**

cout << &array[5] - ptr << endl; **//value address of index 5 subtracted by address of 2 (so 5-2 = 3) printed**

swap1(&array[0], &array[1]);**//addresses of index 0 and 1 are swapped, but not the values in the array**

swap2(array, &array[2]);**//array[0] is now 4, array[2] is now 79**

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

cout << array[i] << endl;**//{4, -1, 79, 9, 22, 19} with a newline after every integer**

}

**Output is**

3

4

79

-1

9

22

19

5. void removeS(char\* msg){

char\* fixedMsg = msg; //created a local pointer that pointers to the msg cstring

while(\*fixedMsg != '\0'){

if(\*fixedMsg == 'S' || \*fixedMsg == 's'){

fixedMsg++;

}

else{

\*msg = \*fixedMsg;

msg++;

fixedMsg++;

}

}

\*msg = '\0';// msg = he'll be a male prince

}

int main()

{

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

removeS(msg);

cout << msg; // prints he'll be a male prince.

}