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 -= 2;

while (ptr < arr+3)

{

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

ptr++;

}

}

1. B.

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;

}

When the main function passes ptr when calling findMax, it passes a pointer but unless it is passed by reference, the changes made to the pointer will not remain. A fix would be to pass pToMax by reference using the & sign.

1. C.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int\* ptr;

int ncubed;

ptr = &ncubed;

computeCube(5, ptr);

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

}

In the main function, ptr doesn’t point to any specific variable, and thus a valid fix would be to create an local int variable and let ptr hold its memory address.

1. D.

// return true if two C strings are equal

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 == 0 && \*str2 == 0); // both ended at same time?

}

int main()

{

char a[15] = "Noor";

char b[15] = "Noah";

if (strequal(a,b))

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

}

The problems with implementation are that the pointers are being compared, where the elements that should be compared are the characters being pointed to by the pointers. A fix would be to add the \* operators in front of the pointers, to refer to the values and not the pointers.

1. E.

In getPtrToArray, the pointer to the first element of anArray is returned but when it initializes the ptr pointer in main, the ptr pointer doesn’t know which object to point to as the scope of the local array anArray does not reach the main function. Additionally, the memory stored by getPtrToArray remains but may be overwritten when calling the function f, where memory allocated for junk overwrites the memory stored for anArray.

double \*cat;

double mouse[5];

cat = mouse+4;

\*cat = 42;

\*(mouse+3) = 25;

cat -= 3;

cat[1] = 17;

mouse[2] = 54;

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

bool d = (cat == mouse);

1. A.

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

{

const double\* ptr = scores;

double tot = 0;

int counter = 0;

while (counter != numScores)

{

tot += \*(ptr+counter);

counter++;

}

return tot/numScores;

}

1. B.

// This function searches through str for the character chr.

// If the chr is found, it returns a pointer into str where

// the character was first found, otherwise nullptr (not found).

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

{

for (int k = 0; \*(str+k) != 0; k++){

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

return str+k;

}

return nullptr;

}

1. C.

// This function searches through str for the character chr.

// If the chr is found, it returns a pointer into str where

// the character was first found, otherwise nullptr (not found).

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

{

while(\*str != 0){

if(\*str == chr)

return str;

str++;

}

return nullptr;

}

#include <iostream>

using namespace std;

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

{

if (\*a > \*b) \\ returns the value that is greater

return a; \\ between the variable pointed to by a

else \\ and the variable pointed to by b

return b;

}

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

{ \\ points a to the object to pointed by

int\* temp = a; \\ b and b points to the object pointed

a = b; \\ to by a.

b = temp; \\ the pointers remain the same

\\ afterward because they are not

} \\ passed by reference.

void swap2(int\* a, int\* b) \\ swaps the value of the variable

{ \\ pointed to by *a* with the value of

int temp = \*a; \\ the variable pointed to by *b*

\*a = \*b;

\*b = temp;

}

int main()

{

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

\\ maxwell returns a pointer to

int\* ptr = maxwell(array, &array[2]); \\ array[0], as 5 > 4

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

ptr += 2; \\ ptr points to array[2]

ptr[1] = 9; \\ sets array[3] to 9

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

cout << &array[5] - ptr << endl; \\ 5-2=3

swap1(&array[0], &array[1]); \\ swap1 does nothing

swap2(array, &array[2]); \\ { -1, 79, 4, 9, 22, 19}

\\ is changed to:

for (int i = 0; i < 6; i++) \\ { 4, 79, -1, 9, 22, 19}

cout << array[i] << endl;

}

3

4

79

-1

9

22

19

int main()

{

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

removeS(msg);

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

}

void removeS(char\* sRemoved)

{

char\* counter = sRemoved;

while (\*sRemoved != 0) {

counter = sRemoved;

while (\*counter == 's' || \*counter == 'S') {

for (; \*counter != 0; counter++) {

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

}

counter = sRemoved;

}

sRemoved++;

}

\*sRemoved = '\0';

}