Homework

1. a.

Original code:

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

while (ptr >= arr)

{

ptr--;

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

}

}

Bugs:

In the first line highlighted, the expression is not assignable. \*ptr is a double rather than a pointer. Therefore \*ptr + 1 makes no sense.

In the while loop, the number will be printed backwards. Also, due to the fact that the line ptr --; comes before the next line, it will start at position index 1 instead of 2, which is not what we want.

Fix:

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; // set pointer to index 0 position

while (ptr <= arr + 2)//print value from index 0 to index2

{

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

ptr++;

}

}

b. Original code

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;

}

Bug:

The pointer is unchanged after we leave the function.

Fix:

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;

}

}

c. Original code:

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int\* ptr;

computeCube(5, ptr);

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

}

Bugs: It is wrong to leave the pointer uninitialized. It has to point to something otherwise it will be null.

Fix:

int main()

{ int a;

int\* ptr = &a; // initialize ptr to point to a

computeCube(5, ptr);

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

}

d. Original code:

// 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 == str2; // 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";

}

Bugs: str1 and str2 are pointers that point to elements in different arrays. It does not make sense to compare the two. The program is intended to compare the char that these pointers point to. Therefore, the comparison should be made between \*str1 and \*str2

Fix:

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?

}

e. Original code:

#include <iostream>

using namespace std;

int\* getPtrToArray(int& m)

{

int anArray[100];

for (int j = 0; j < 100; j++)

anArray[j] = 100-j;

m = 100;

return anArray;

}

void f()

{

int junk[100];

for (int k = 0; k < 100; k++)

junk[k] = 123400000 + k;

}

int main()

{

int n;

int\* ptr = getPtrToArray(n);

f();

for (int i = 0; i < 3; i++)

cout << ptr[i] << ' ';

for (int i = n-3; i < n; i++)

cout << ptr[i] << ' ';

cout << endl;

}

Bug: The array is created in the function. It will be erased (with random value stored in it) after the program is done with the function. Therefore, it will be incorrect for the pointer to points to the value in this array.

2. a. double a;

double\* cat = &a;

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)

{

const double\* ptr = scores;

double tot = 0;

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

{

tot += \*(ptr + 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( char\* str, char chr)

{

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

if (\*str == chr)

return str;

return nullptr;

}

4. code :

#include <iostream>

using namespace std;

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

{

if (\*a > \*b)

return a;

else

return b;

}

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

{

int\* temp = a;

a = b;

b = temp;

}

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

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

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

// set ptr to &array[0], because 5(element at position 1) is greater than 4 (element at position 2)

int\* ptr = maxwell(array, &array[2]);

// set the element at position 0 to -1

\*ptr = -1;

// move ptr to position 2

ptr += 2;

// set array[3] to 9

ptr[1] = 9;

// set array[1] to 79

\*(array+1) = 79;

// calculate how far is ptr from &array[5]

cout << &array[5] - ptr << endl;

// swap the two pointers &array[0] and &array[1]

swap1(&array[0], &array[1]);

// swap the int’s stored in addresses pointed to by array and &array[2]

swap2(array, &array[2]);

// print out the elements in the array from left to right(each occupies a single line)

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

cout << array[i] << endl;

}

Output: 3

4

79

-1

9

22

19

5.

void removeS (char\* msg)

{

char\* dp = msg;

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

if ( \*msg != 's' && \*msg != 'S')

{

\*dp = \*msg;

dp++;

}

\*dp = '\0';

}