CS 31 Assignment 6

Edward Chu 504772148

1a.

Bugs:

1. \*(ptr + 1) -> should have parenthesis
2. ptr should be reset to the first element before loop
3. ptr should be incremented after \*ptr is printed instead

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++;

}

}

1b.

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;

}

This is because the original function only creates a copy of pToMax and alters only the copy of the pointer, but never the original pToMax. We can fix this by referring to the actual pointer instead of creating a copy of it, using int\*& pToMax instead.

1c.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int a;

int\* ptr = &a;

computeCube(5, ptr);

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

}

The value of the pointer is not initialized, so using the dereference operator on it will cause a runtime error. We can fix this by initializing it to point to some integer.

1d.

// 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] = "Zhao";

char b[15] = "Zhou";

if (strequal(a,b))

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

}

The program is comparing between the pointers, instead of the characters pointed to by the pointer. Thus, the dereference operator should be applied when comparing the pointers.

1e.

For the first function, the int anArray[100] is a local variable, thus will be deleted and the memory location of it will be used for other things when the function is completed. Therefore, returning a pointer to this array will only print out garbage values.

2a. double\* cat;

2b. double mouse[5];

2c. cat = &mouse[4];

2d. \*cat = 25;

2e. \*(mouse+3) = 42;

2f. cat -= 3;

2g. cat[1] = 54;

2h. cat[0] = 17;

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

2j. bool d = (cat == mouse);

3a.

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

{

double tot = 0;

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

tot += \*(scores + i);

return tot/numScores;

}

3b.

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

{

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

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

return str+k;

return nullptr;

}

3c.

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

{

while(\*str != 0){

if (\*str == chr)

return str;

str++;

}

return nullptr;

}

4.

#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 };

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

//ptr points to array[0], since \*array is 5 and \*&array[2] is 4

\*ptr = -1;

//array[0] is now -1

ptr += 2;

//ptr now points to array[2]

ptr[1] = 9;

//array[3] is now 9

\*(array+1) = 79;

//array[1] is now 79

//array is now {-1, 79, 4, 9, 22, 19}

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

//Prints out the difference of the two pointers, which is 3, as ptr points to array[2]

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

//Does nothing! Only modifies the copy of the pointer created

swap2(array, &array[2]);

//Swaps the position of array[0] and array[2]

//array is now {4, 79, -1, 9, 22, 19}

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

cout << array[i] << endl;

}

Output of Q4:

3

4

79

-1

9

22

19

5.

void removeS(char\* msg){

char\* ptr = msg;

//for each char in the msg

while(\*ptr != 0){

//if the char pointed to by ptr is ‘s’ or ‘S’

if(toupper(\*ptr) == ‘S’){

//copy all other chars after it and shift them by 1 position

while(\*ptr != 0){

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

ptr++;

}

ptr=msg-1; //start over

}

ptr++;

}

}