Nathan Tjoar

005081232

Lecture 2

Homework

1a.

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++;

while (ptr > arr)

{

ptr--;

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

}

}

1b. pToMax should be a call by reference parameter, otherwise, the function will not change the value of 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;

}

}

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;

}

1c. The pointer is not initialized, leading to undefined behavior.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int i;

int\* ptr = &i;

computeCube(5, ptr);

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

}

1d. In this function, str1 and str2 are initialized as pointers, so when str1 != 0 is asked or str2 != 0 is asked, it is asking if the pointer is the same as the null pointer. Similarly when comparing them and returning a value, it is asking if the addresses are equal.

// 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] = "Zhou";

char b[15] = "Zhou";

if (strequal(a,b))

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

}

1e. The function getPtrToArray erases storage for anArray, which yields undefined behavior when ptr[i] is called.

2.

int main()

{

double\* cat;

double mouse[5];

cat = &mouse[4];

\*cat = 25;

\*(mouse + 3) = 42;

cat -= 3;

cat[1] = 27;

cat[0] = 54;

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

bool d = (cat == mouse);

}

3a.

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

{

int k = 0;

double tot = 0;

while (k != numScores)

{

tot += \*(scores + k);

k++;

}

return tot/numScores;

}

3b.

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.

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

{

while(\*str != 0)

{

if (\*str == chr)

{

return str;

}

str++;

}

return nullptr;

}

4.

Prints:

3

4

79

-1

9

22

19

Reasons:

Maxwell is called with parameters array and array[2], which returns &array[0]

&array[0] is then replaced with the value -1

pointer is then incremented to point to array[2]

array[3] is replaced with the value 9

array[1] is replaced with the value 79

When swap1 is called, the function does nothing other than swapping its copies of the pointers

Swap2 is then called and swaps the values of array[0] and array[2]

5.

void removeS(char\* str)

{

char\* dest = str;

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

{

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

{

\*dest = \*str;

dest++;

}

str++;

}

\*dest = '\0';

}