Miao Hu

**Project 6**

**1a)**

int main()

{

int arr[3] = { 5, 10, 15 };

int\* ptr = arr;

\*ptr = 30; // set arr[0] to 30

\*ptr + 1 = 20; // **BUG: ptr + 1 should be enclosed in parentheses**

ptr += 2;

ptr[0] = 10; // **BUG: should just be \*ptr**

while (ptr >= arr) // **BUG: prints in reverse order**

{

ptr--;

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

}

}

/////////////My solution/////////////

int main()

{

int arr[3] = { 5, 10, 15 };

int\* ptr = arr;

\*ptr = 10; // set arr[0] to 10

\*(ptr + 1) = 20; // set arr[1] to 20

ptr += 2;

\*ptr = 10; // set arr[2] to 30

ptr = arr; // reset ptr to first element of arr

while (ptr >= arr)

{

ptr--;

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

}

}

**1b)**

The function will not behave as expected because of a bug within findMax's parameters. The argument: int\* pToMax is passed by value, which means create a local copy of the pointer passed to findMax. Any changes made to pToMax won't change the value when the function returns. To fix this, **change the argument to be passed by reference instead of value**:

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;

}

}

}

**1c)**

The main function has a problem is that the pointer doesn't initialized to any memory address. As a result, a compilation error. To fix this, I declared an int var and initialized it to some arbitrary value, then assigned ptr to the address of var.

int main()

{

int var = 0;

int\* ptr = &var;

computeCube(5, ptr);

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

}

**1d)**

Instead of comparing the characters at the memory addresses that str1 and str2 currently point to, the while condition only compares the memory address. The if statement and return condition also run into the same problem. To fix this, I dereferenced the pointers with '\*' when str1 and str2 were compared.

bool strequal(const char str1[], const char str2[])

{

while (\*str1 != 0 && \*str2 != 0)

{

if (\*str1 != \*str2) // compare corresponding characters

return false;

str1++;

str2++; // advance to the next character

}

return (\*str1 == \*str2); // both ended at same time?

}

**1e)**

When the function getPtrToFunction returns, anArray is a variable local to the function and is thus cleared from memory. As a result, the values that exist at the memory address pointed to by ptr are undefined. When the function f is called, it can also assign garbage values in the memory spaces previously allocated to anArray.

**2)**

1. double\* cat;
2. double mouse[5];
3. cat = &mouse[4];
4. \*cat = 25;
5. \*(mouse + 3) = 54;
6. cat -= 3;
7. cat[1] = 42;
8. cat[0] = 17;
9. bool d = (cat == mouse);
10. bool b = (\*cat == \*(cat + 1));

**3a)**

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

{

const double\* ptr = scores;

int count = 0;

double tot = 0;

while (count < numScores)

{

tot += \*(ptr + count);

count++;

}

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)**

// return the pointer which has larger value

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

{

if (\*a > \*b)

return a;

else

return b;

}

// do nothing, just swap the pointer's adress

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

{

int\* temp = a;

a = b;

b = temp;

}

// swap value

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]);

// Compares values (5,4), and returns the memory address of which one is

// greater. 5 > 4, so assigns ptr to point at arr[0]

\*ptr = -1; //Set arr[0] = -1

ptr += 2; //Set ptr pointing at arr[2]

ptr[1] = 9; //Set arr[3] = 9

\*(array+1) = 79; //Set arr[1] = 79

//current array = {-1, 79, 4, 9, 22, 19}

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

//prints ((array + 5)–(array + 2)) which is equal to 3, Newline after 3

swap1(&array[0], &array[1]); // nothing actually happen

//swaps memory address of pointers a & b

swap2(array, &array[2]); //swaps values at the 0 (-1), and 2 (4)

//current array = {4, 79, -1, 9, 22, 19}

for (int i = 0; i < 6; i++) //prints every element of array perline

cout << array[i] << endl;

}

**Output:**

3 //from (&array[5] – ptr) = 3

4 //swapped with -1

79 //from \*(array+1) = 79

-1 //swapped with 4

9 //from ptr[1] = 9

22 //unchanged

19 //unchanged

**5)**

void removeS(char\* ptr)

{

char\* tempPtr = ptr;

while (\*ptr ！= 0)

{

if (\*ptr == 's' || \*ptr == 'S')

{

tempPtr = ptr; // set temp pointer to current 's'pointed by ptr

while (\*tempPtr)

{

\*tempPtr = \*(tempPtr + 1); //shift all chars one place left

tempPtr++;

}

continue; // break the inner loop to check next character

}

ptr++; // Check next char when current char isn't 's'

}

}