Miko Farin (704789591)

CS 31

Project 6

1.

a. Bugs in program:

1. Due to the operators’ order of precedence, the line, \*ptr + = arr [0] \* 10; , is not correctly written. In the code provided, writing the code , \*ptr + = arr [0] \* 10; is wanting to do (\*ptr) + 1 = arr [ 0 ] \* 10; which evaluates to the int 2 and not the value of 1. The code should be corrected to \*(ptr + 1) = arr [0] \* 10;. Using the parentheses allows us to tell the computer on what we want.
2. The second bug is that the while loop does not access arr [2]. The problem is that it excludes the first array and goes to arr[-1] .

Fixed code:

int main()

{

int arr[4] = { 0, 1, 2, 3 };

int\* ptr = arr;

\*ptr = arr[1]; // set arr[0] to 1

**\*(ptr + 1) = arr[ 0 ] \* 10;** // set arr[1] to 10

ptr += 2;

\*ptr = arr[1] \* 10; // set arr[2] to 100

ptr[1] = 1000; // set arr[3] to 1000

**ptr += 2;**

while (ptr > arr)

{

**ptr--;**

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

}

cout << endl;

return(0);

}

b. The problem with the code is that the function consists of the copying of the correct value of the pointer ptr. The function is supposed to change the actual value of the ptr and not just copy the pointer. We need to change it to be passed by reference and not just by value.

Fixed code:

void findLastZero(int arr[], int n, **int\* &p**)

{

p = nullptr; /// default value if there isn't a 0 in the array at all

for (int k = n - 1; k >= 0; k--)

{

if (arr[k] == 0) // found an element whose value is 0

{

p = arr + k; // change the value of p

break; // stop looping and return

}

}

}

int main()

{

int nums[6] = { 10, 20, 0, 40, 30, 50 };

int\* ptr;

findLastZero(nums, 6, ptr);

if (ptr == nullptr)

{

cout << "The array doesn't have any zeros inside it." << endl;

}

else

{

cout << "The last zero is at address " << ptr << endl;

cout << "It's at index " << ptr - nums << endl;

cout << "The item's value is " << \*ptr << " which is zero!" << endl;

}

return( 0 );

}

c. The problem of the code is the declaration of the pointer. It creates a pointer without being initialized properly. This sends an uninitialized value to the function.

Fixed code:

void biggest(int value1, int value2, int \* resultPtr)

{

if (value1 > value2)

{

\*resultPtr = value1;

}

else

{

\*resultPtr = value2;

}

}

int main()

{

**int i;**

**int \* p = &i;**

biggest(15, 20, p);

cout << "The biggest value is " << \*p << endl;

return(0);

}

d. During the comparison of the strings, they forget to use ‘\0’ to be able to access the characters within the array.

Fixed Code:

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

{

bool result = true;

while **(\*str1 != 0 && \*str2 != 0)** // zero bytes at ends

{

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

{

result = false;

break;

}

str1++; // advance to the next character

str2++;

}

if (result)

{

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

}

return(result);

}

int main()

{

char a[10] = "pointy";

char b[10] = "pointless";

if (match(a, b))

{

cout << "They're the same!" << endl;

}

}

e. After the function is called, the array created within that function does not exist outside the function when it returns to the main function. The pointer will continue, but it will five a value that is in that pointer’s storage.

2.

1. string \* fp;

**f. declares a pointer variable to point to a variable of type string**

2. string fish[ 5 ];

**g. declares a five element array of string**

3. fp = &fish[ 4 ];

**a.sets a pointer variable to the last element of an array of five strings**

4. \*fp = "salmon";

**b. sets the string pointed by a pointer to by a pointer variable to the value “salmon”**

5. fp -= 3;

**d. moves the pointer fp back three strings in the array it points to**

6. \*(fp + 3) = "salmon";

**c. sets the fourth element of an array pointed to by the variable fp to the value of “salmon”**

7. fp[ 0 ] = "salmon";

**b.sets the string pointed to by a pointer variable to the value “salmon”**

8. bool b = (fp == fish);

**e. initializes a boolean to true if the pointer variable fp points to the string at the start of the fish array, false otherwise**

9. bool b = (\*fp == \*(fp + 1));

**h. initializes a boolean to true if fp points to a string whose value matches the string immediately following the string pointed to by fp,**

**false otherwise**

3.

Line 1: creates an array of 6 elements, consisting of the integers {5, 3 ,4 ,17, 22, 19}

Line 2: Calls the function minimart, returns a pointer and is set to int ptr.

a.The function takes two parameters or &arr [0] and &arr[2]. The function compares the two. Since array [0] = 5 and array [2] = 4, the function returns &array[2] and is stored in ptr

Line 3: assigns ptr[1] = 9 or array [3] = 9

Line 4 : moves ptr up by two or ptr +=2 sets &arr[4]

line 5: \*(ptr + 2) = -1, meaning it sets array [4] = -1

line 6: \*( array+1) = 79, sets array[1] to 79

line 7: finds the difference between &array[5] - &array{4]; this is equal to 1

line 8: Takes into array into the function swap, &array [0] and &array[1]

a.&array [0] holds 5 and &array [1] holds 79 due to lines 6.Then we set temp integer pointer variable held at &array [0]. This makes the address held at &array [0] to be the address that is held at &array [1]. THe address held at &array [1 ] is now temp pointer variable which is &array [0].

Line 9: The swap2 function takes in two pointers as parameters.array which is equivalent to 5 and &array [2] which holds 4. This function swaps the ints pointed at.

array values:

array [0] = 4

array [1]= 79

array [2]= 5

array [3]= 9

array [4]= -1

array [5]= 19

Program prints:

diff =1 (lines 2 and 5)

4 (lines 6 and 7

79 (lines 4 and 6)

5 (line 7)

9 (line 1)

-1 (lines 2 and 3)

19

4.

Code:

#include <iostream>

void deleteCapitals(char\* GivenMessage) {

char\* pointer = GivenMessage;

while (\*pointer != '\0') {

if (\*pointer >= 'A' && \*pointer <= 'Z') {

//creates the complement of desired condition

}

else { // if it is not a capital

\*GivenMessage = \*pointer; ++GivenMessage;

}

++pointer; }

\*GivenMessage = '\0'; }

int main()

{

char msg[100] = "Happy Days Are Here Again!";

deleteCapitals(msg);

cout << msg << endl; // prints: appy ays re ere gain!

}