Terry Ye ID:004757414

**Problem 1:**

**A:**

There are three bugs in the original function. The first is that the \*ptr+1 is just value 1 and arr[1] is \*(ptr+1).Second is that it misses ptr++ before while. So there will only be three values printed out. Third is that in the while loop, the “ptr--“ is before the print code, so it will omit the last element of the array and only print out the first 3 elements of the array.

int main()

{

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

int\* ptr = arr[1]

\*ptr = arr[ 1 ]; // set arr[0] to 1 {1,1,2,3}

\*(ptr + 1) = arr[ 0 ] \* 10; // set arr[1] to 10 {1,10,2,3}

ptr +=2;

ptr[0] = arr[ 1 ] \* 10; // set arr[2] to 100

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

ptr++;

while (ptr >= arr)

{

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

ptr--;

}

cout << endl;

return( 0 );

}

**B:**

The problem is that the findLastZero function is a pass-by-value function so the pointer p is not changed and will always equal nullptr, so we just need to change the function to a pass-by-reference function for pointer p.

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

**C:**

The probllem is that the pointer p is not initialized so the function has an undefined behavior.

We should initialzed the pointer p to a value:

int\* p=new int;

**D:**

while (str1 != 0  &&  str2 != 0)

if（str1 != str2）: these code compares the pointer and it is always not same. It should compare the

value it pointed at instead

Change it to while (\*str1 != ‘\0’  &&  \*str2 != ‘\0’)

if(\*str1 != \*str2)

result = (str1 == str2): this always returns false, so to evaluate if it

ends at same time. We should test if \*str1 and

\*str2 equals 0 after the while function

if(result)

{

result=(\*str1==’\0’ && \*str2== ‘\0’);

}

**E:**

arr[k]= fibonacci[k] starts from fibonacci[0] instead of fibonacci[1].

The function computeFibonacciSequence returns the address of an array and cannot be processed correctly.

**Problem 2:**

1-f 2-g 3-a 4-b 5-d 6-c 7-b 8-e 9-h

**Problem 3:**

It prints

diff=1:

4:

79

5

9

-1

19

First line: because minimart(array,&array[2])compares array[0] and array[2] and sets ptr=&array[2], and then ptr+=2 sets ptr= &array[4]. So &array[5]-ptr=1

Rest: ptr=&array[2] ptr[1]=9 sets array[3]=9

Ptr+=2 changes ptr=& array[4]. \*ptr=-1 sets array[4]= -1.

\*(array+1)=79 sets array[1]=79

swap1(&array[0], &array[1])doesn’t change any value because swap1 is a pass-by-

value function and the parameters remain the same.

swap2(array, &array[2] swaps array[0] and array[2] because the parameter is the pointer but the

function swaps the value the pointer points to, not the pointer itself. So the function works and

swaps the array[0] and array[2] because they are the values pointed by array and &array[2]/

The array now is{4, 79, 5,9,-1,19}

**Problem 4:**

void deleteCapitals(char\* s)

{

char\* a = s;

while (\*a != ‘\0’)

{

if (isupper(\*a)) { //if is

a++;

continue;

}

else

\*s = \*a;

a++;

s++;

}

while (\*s != ‘\0’)

{

\*s = ‘\0’; //change the value left at the end to null

}

}