Workshop 04

Abcedi Ilacas

04/02/24

SFT221NFF

Bugs

1. Original :

int factorial(const int n)

{

return (n <= n) ? n \* factorial(n - 1) : 1;

}

Fixed :

int factorial(const int n)

{

return (n > 1) ? n \* factorial(n - 1) : 1;

}

Code Issue :

The (n <= n) condition in this function was always going to be true. I used (n > 1) instead to correct the line and properly recurse until n equals 1.

Technique :

I basically just used logical analysis of the condition. I could tell it did not make sense and will never change.

1. Original :

int reduceFactorial(const int n)

{

return n / n;

}

results.results[i] = reduceFactorial(results.results[i]);

Fixed :

Removed function and the line in main from the program.

Code Issue :

This function is basically useless as it will ALWAYS return 1 since n does not change in the function.

Technique :

Looking at this function mathematically showed it doesn’t have a useful purpose.

1. Original :

void computeFactorials(struct FactorialResults results, int numFactorials)

Fixed:

void computeFactorials(struct FactorialResults \*results, int numFactorials)

{

int i;

for (i = 0; i < numFactorials; i++)

{

results->results[i] = factorial(i);

}

results->numResults = numFactorials;

}

Code Issue :

The original was using pass by value instead of pass by reference which made it so it wont update the structure. I corrected it by using a pointer.

Technique :

I learned in IPC144 about pass by value and pass by reference and their importance. Pass by reference will update the variable outside the function.

1. Original :

Int main(){

….

printf("%5d %12f\n", i, results.results[i]);

….

}

Fixed :

printf("%5d %12d\n", i, results.results[i]);

Issue :

The issue in this line is the use of the wrong printf specifier. It was originally using %f for an int so I corrected it to %d for int formatting.

Technique :

I recognized the mismatch between the data type and format specifier.

1. Original :  
   computeFactorials(results, NUM\_FACTS);

Fixed :

computeFactorials(&results, NUM\_FACTS);

Issue :

The struct was not being changed die to it being passed by value originally. I fixed it by passing the address of results.

Technique :

I saw and identified the lack of changes to results after the function was being called.

1. Original :

#define NUM\_FACTS 100

Fixed :

#define NUM\_FACTS 34

Issue :

The output has a limit and will not go past the 33rd factorial. So its not necessary to have 100 NUM\_FACTS.

Technique :

By running the code multiple times, it is clear that it will never calculate pass 33 since of the overflow in C.