SFT221 – Workshop 4

Learning Outcomes

- Recognize common bugs,
- Fix common bugs,
- Learn debugging techniques.

Instructions

The following program is supposed to print a list of integers and the factorial of one less than the number on the line beside it. Unfortunately, some bugs have crept into the program. You should:

- Run the program and see how it works,
- Look for bug clues that were covered in the lectures,
- Use whatever debugging techniques you want to find the bugs,
- Fix the bug(s),
- Repeat steps above until all bugs are fixed.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

#define MAX_FACTORIALS 10000
#define NUM_FACTS 100

struct FactorialResults
{
	int results[MAX_FACTORIALS];
	int numResults;
};

int factorial(const int n)
{
	return (n <= n) ? n * factorial(n - 1) : 1;
}

int reduceFactorial(const int n)
{
	return n / n;
}

void computeFactorials(struct FactorialResults results, int numFactorials)
{
	int i;

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

```
	{
		results.results[i] = factorial(i);
	}

	results.numResults = numFactorials;
}

int main(void)
{
	struct FactorialResults results = { {0}, 0 };
	int i;

	computeFactorials(results, NUM_FACTS);
	for (i = 0; i < NUM_FACTS; i++)
	{
		results.results[i] = reduceFactorial(results.results[i]);
		printf("%5d %12f\n", i, results.results[i]);
	}

	return 0;
}
```

Deliverables

Due Date:

This workshop is due 2 days after your lab day. Late workshops will not be accepted.

You should submit:

- A zipped Visual Studio project that contains the debugged, working version of the code,
- A document which lists:
  - The line(s) containing each bug,
  - The corrected version of the line(s),
  - What was wrong with the line(s) and how you fixed it,
  - The technique you used to recognize and find the bug.
- A document called reflect.txt which answers the reflections below.

A Reflection, Research and Assessment

Reflections should consider the questions in depth and not be trivial. Some reflections might require research to properly answer the question. As a rough guideline, the answer to each reflection questions should be about 100 words in length.

1. What was the most useful technique you used to find the bugs? Why was it more useful than other techniques you tried?
2. Look up answers to the following questions and report your findings:
   a. What are the largest integer and double values you can store?
   b. Why is there a limit on the maximum value you can store in a variable?

c. If you exceed the maximum value an integer can hold, what happens? Explain why the format causes this to happen.
d. What is the format for the storage of a floating point variable? How does this differ from the way an integer is stored?

3. What is the default amount of stack memory that is given to a program when Visual Studio starts a C or C++ program? What is the default heap size? Did you hit any of the limits? If so, which one(s)? If you hit a limit, would increasing the amount of memory allocated to the program fix the problem? Justify your answer. Why do they limit the stack and heap size for a program?

Marking Rubric

Number of bugs found	15%
Number of bugs correctly fixed	15%
Quality of explanation of each bug	15%
Reflection 1	15%
Reflection 2	20%
Reflection 3	20%