**CPSC 331 Assignment#2**

Methodology***:***

        The part one of my program takes two inputs from user. It requires no explanation since it is trivial.

        The part two and part three is a nested loop for calculating intermediate values in the beginning of  the loop in part two, I set the count ‘a’ equals to the length of second user input, as long as a is greater than and equal to 0, count ‘a’ will subtract itself by 1 at every end of the loop. This outer loop will help me to go over every digit in the first input in order to multiply by the digits in second input which I will explain in the following paragraph.

        Before the inner loop start, there is another loop inside the outer loop which add zero(s) after the first iteration (the first intermediate calculation do not need to add zeros). In the inner loop, I cast the first character of first and second inputs into short numbers and multiply them. The if statement checks the result of multiplication is greater than or equal to 10, I initialize the flag to false (multiplication is less than 10) so that the if statement will be executed in the first iteration. If multiplication is less than 10, I add the multiplication result to the carry from last iteration. Else if, the multiplication is greater than 9, that means there is carry for the multiplication. Since the multiplication of two digits’ range is from 10 to 81 if the result is larger than 9, so that I take the last digit as the result of the iteration and save the first digit as the carry for the coming iteration. If the multiplication in first iteration is greater than 9, the “else if” will be executed instead of “if” when running the second iteration. The purpose of having if and else-if construction in the loop is that I could check if the multiplication plus the carry will again bigger than 9 or not. After the first index of second input has multiplied the entire digits in the first input, the first intermediate calculation has finished. Before incrementing the loop counter and entering the loop of second digits of the second input, I save the intermediate calculation into the array as strings. After the nest loop has finished, all the intermediate calculations will be saved inside the array.

        The part four adds one intermediate calculation with the result by using the add (string, string) method. I will explain the add method in the next paragraph. After the loop, I print out all the intermediate calculations which is very trivial.

        There are two parts in the “add” function, the first part will make shorter length intermediate calculation the same as the longer length intermediate calculation by adding zeros before it. After the algorithm, two intermediate calculation will have the same length. Second part of “add” function will do the addition for two intermediate calculations. I start with counter ‘g’ which is equal to the last index of one intermediate calculation.(it does not matter which one since two intermediate calculations are in same length) As long as the counter ‘g’ is greater than or equal to the last index of the number, the loop will execute. First, I convert two character from two numbers at index of ‘g’ into short, and then I add the two numbers together. The if statement checks the result is greater than 9, I initialize the flag to false (addition is less than 10) so that the if statement will be executed in the first iteration. If addition is less than 10, it means we have no carry. If addition is greater than 9, I subtract total by 10 in order to get the result without carry. Then I combine the string result from last iteration with the result in this iteration. . If the addition in first iteration is greater than 9, the “else if” will be executed instead of “if” when running the second iteration. The purpose of having if and else-if construction in the loop is that I could check if the addition plus the carry will again bigger than 9 or not. At the end of the loop, the sum of two intermediate calculations will be returned.

Correctness of algorithm and termination:

**Part 1: Ask for user inputs**

Takes user inputs. This is very trivial and requires no proof for it. No termination code required

**Part 2: Calculate intermediate values(outer loop)**

{P}

N: positive string no negative or foreign symbol

The numbers in strings are not less than 1 or foreign symbol

2. 0 <= short 'a' < secondInput.length

3. 0 <= short 'd' < zero

{Q}

The Strings have not been changed

The intermediate values have been written into the List

For(a: secondInput.length){p}

For(d: zero){p}

        End{q}

End{q}

{P} S {Q} since no matter what occurs a statement ill always happening concatenating to the string which is our Q and the for loop will eventually end when there are no more items in the string.

Therefore the loop will run (ath + 1)\*(dth + 1) times since there are no loop variants that affect the loop whatsoever and the preconditions and postconditions are met since every {P} corresponds to a {Q}

**Part 3: Calculate intermediate values(inner loop)**

        {P}

The numbers in strings are not less than 1 or foreign symbol

0 <= short 'b' < firstInput.length

{Q}

The strings have not changed

Boolean carry = false

For(b : firstInput){p}

If(false)

If(total > 9)

         True{Q}

        If(total < 10)

                    False{Q}

        If(total > 9)

                    True{Q}

Else(! false)

        If(total <  10)

                    False{Q}

        If(total > 9)

                    True{Q}

{P} S {Q} since no matter what occurs a statement ill always happening concatenating to the string which is our Q and the for loop will eventually end when there are no more items in the string.

Therefore the loop will run bth + 1 times since there are no loop variants that affect the loop whatsoever and the preconditions and postconditions are met since every {P} corresponds to a {Q}

**Part 4: Add the intermediate calculations together**

{P}

1. 0 <= short 'f' < intermediateList.size

{Q}

string result for addition of all intermediate values

For(f : intermediate){p}

End{q}

{P} S {Q} since no matter what occurs a statement ill always happening concatenating to the string which is our Q and the for loop will eventually end when there are no more items in the string.

Therefore the loop will run fth + 1 times since there are no loop variants that affect the loop whatsoever and the preconditions and postconditions are met since every {P} corresponds to a {Q}

**Part 5: Print out all the intermediate calculation**

{P}

1. 0 <= short 'c' < intermediateList.size

{Q}

Output include:

count: short

intermediate calculations: string

For(c : intermediate){p}

End{q}

{P} S {Q} since no matter what occurs a statement ill always happening concatenating to the string which is our Q and the for loop will eventually end when there are no more items in the string.

Therefore the loop will run cth + 1 times since there are no loop variants that affect the loop whatsoever and the preconditions and postconditions are met since every {P} corresponds to a {Q}

**Part 6: Making two intermediate calculations in the same length**

{P}

1. 0 <= short 'c' < intermediateList.size

{Q}

Output include:

count: short

intermediate calculations: string

If(str < result)

for(str : result)

End{q}

else if(str > result)

for(result : str)

End{q}

{P} S {Q} since no matter what occurs a statement ill always happening concatenating to the string which is our Q and the for loop will eventually end when there are no more items in the string.

**Part 7: Addition of two numbers**

{P}

We are taking two numbers in string form

0 <= short 'g' < str.length

{Q}

Outputs single string

Adds the numbers

Boolean carry = false

For(i : str){p}

If(false)

If(total > 9)

         True{Q}

        If(total < 10)

                    False{Q}

        If(total > 9)

                    True{Q}

Else(! false)

        If(total <  10)

                    False{Q}

        If(total > 9)

                    True{Q}

{P} S {Q} since no matter what occurs a statement ill always happening concatenating to the string which is our Q and the for loop will eventually end when there are no more items in the string.

Therefore the loop will run ith + 1 times since there are no loop variants that affect the loop whatsoever and the preconditions and postconditions are met since every {P} corresponds to a {Q}.

WORST CASE SCENARIOS:

**Part 1: Ask for user inputs**

Worst case is 5 lines one for each line of code ran since I am creating variables.

**Part 2: Calculate intermediate values(outer loop)**

**Part 3: Calculate intermediate values(inner loop)**

Worst case is secondInput.length() for the for statements and then inside the for is two for loops variant and so secondInput,length \* (zero - 1) is the worst case.

**Part 4: Add the intermediate calculations together**

Worst case is intermediate.size()

**Part 5: Print out all the intermediate calculation**

Worst case is intermediate.size()

**Part 6: Making two intermediate calculations in the same length**

Worst case is (result.length() - str.length) or (str.length() - result.length)

**Part 7: Addition of two numbers**

Worst case is str.length()