Mariam Tsirekidze ID:823460489

Assignment 2

Report

Problem:

In this assignment I have to create the program which can hold extremely large integer numbers, much bigger than c# can hold and be able to add, subtract and multiply that large numbers. Also the input of numbers should be the text file and my program should be able to read these file and make operations.

Solution:

Infint class:

- The class has two variables: 1) List of integers, which can hold extremely large number, and

  2) bool variable to determine the sign of the number (positive or negative)

SetInfInt(String line):

- The method writes the provided string into List of integers.

GetInfInt():

- The method returns the value of large number as a list of integers

InfInt(string line):

- The constructor initializes the sign of the number, whether it is positive or negative and also rewrites provided string into the list of integers, initializes the large number.

InfInt():

- The default constructor initializes the sign and the number of InfInt when it's created.

Icomparable Inteface and CompareTo(InfInt other):

- The interface and the method helps to understand which InfInt operand's absolute value is larger.

Addition(InfInt other):

- The method adds two List of integers as they are large numbers. The addition starts from the very last digits of the operands, if their sum is greater than 9 then we have carry which should be added to the next couple of digits. The method returns the answer as string.

Substraction(InfInt other):

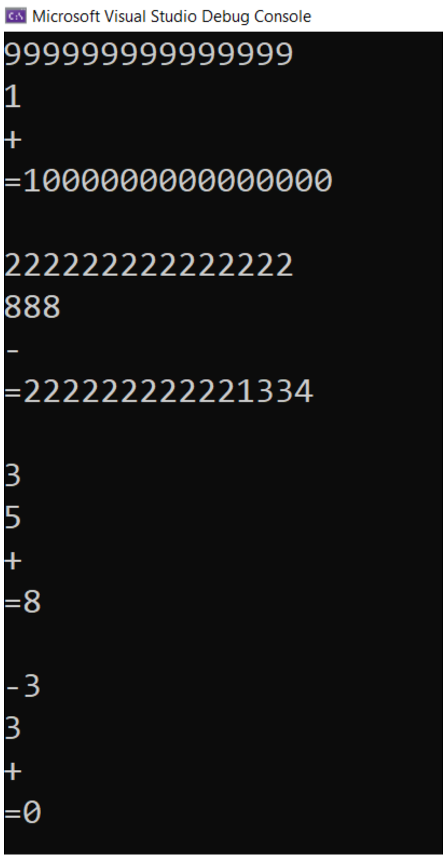
- The method subtracts two List of integers as they are large numbers. First, we should determine which operand is more then other whith the help of CompareTo() method. Then subtract small number from the greater one. The subtraction starts from the very last digits of the operands, if the difference is less then 0 we have carry which should be subtracted from the next couple of digits. The method returns the answer as string.

Multiplication(InfInt other):

- The method multiplies two List of integers as they are large numbers. The multiplication starts from the very last digit of the shorter number. Shorter number's very last digit is multiplied by the longer numbers digits (if the multiplication of couples are >0 we have carry that we should add to the next couple of digits) and large number is saved in the array of string. Every short number's digits multiplied by the greater number is saved in array of strings now we have to add the array items and that is the answer. The method returns the answer as list of integers.

Program class:

First of all the program class reads the text file as the array of strings. First two items in the array are the operands and the third one is operator. So Firstly, I will determine the operator whether it is "-", "+" or "*" and then determine of the signs of the operands in order to call the specific method of the InfInf class (addition, multiplication or subtraction). Pass the string to the specific method and output the answer of the operation.

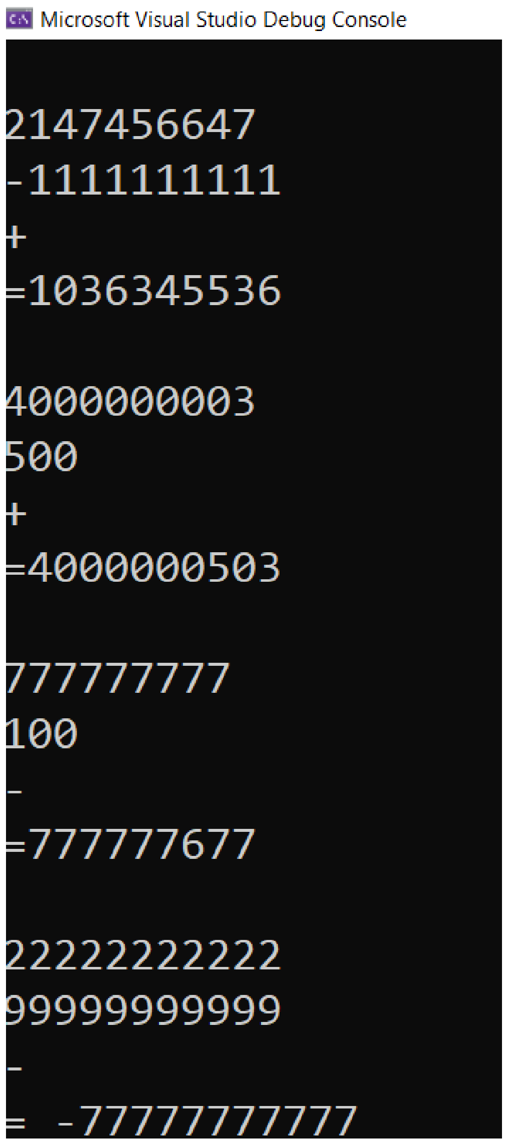

Microsoft Visual Studio Debug Console

```
999999999999999
1
+
=1000000000000000

222222222222222
888
-
=222222222221334

3
5
+
=8

-3
3
+
=0
```

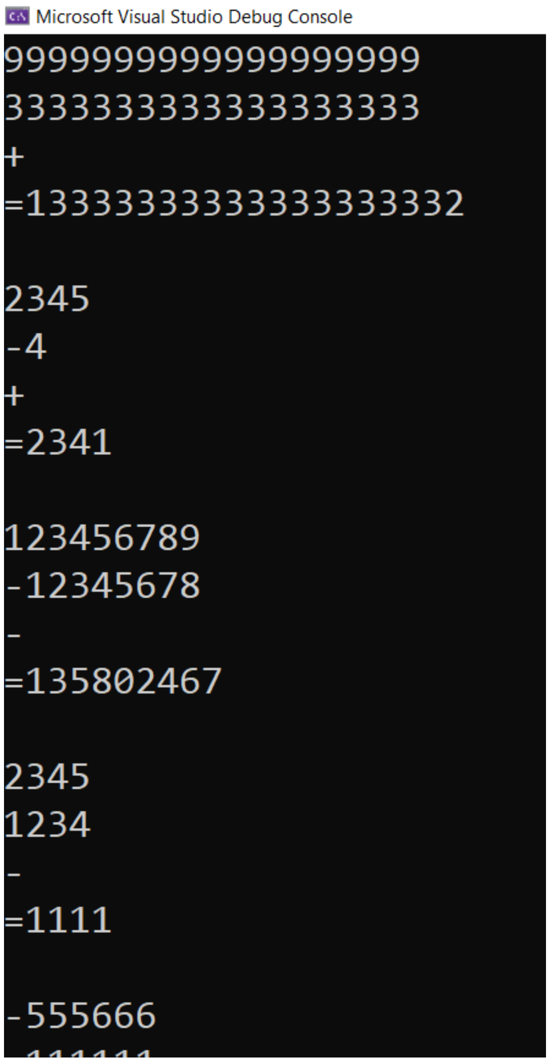

Microsoft Visual Studio Debug Console

```
2147456647
-1111111111
+
=1036345536

4000000003
500
+
=4000000503

777777777
100
-
=777777677

22222222222
99999999999
-
= -77777777777
```



Microsoft Visual Studio Debug Console

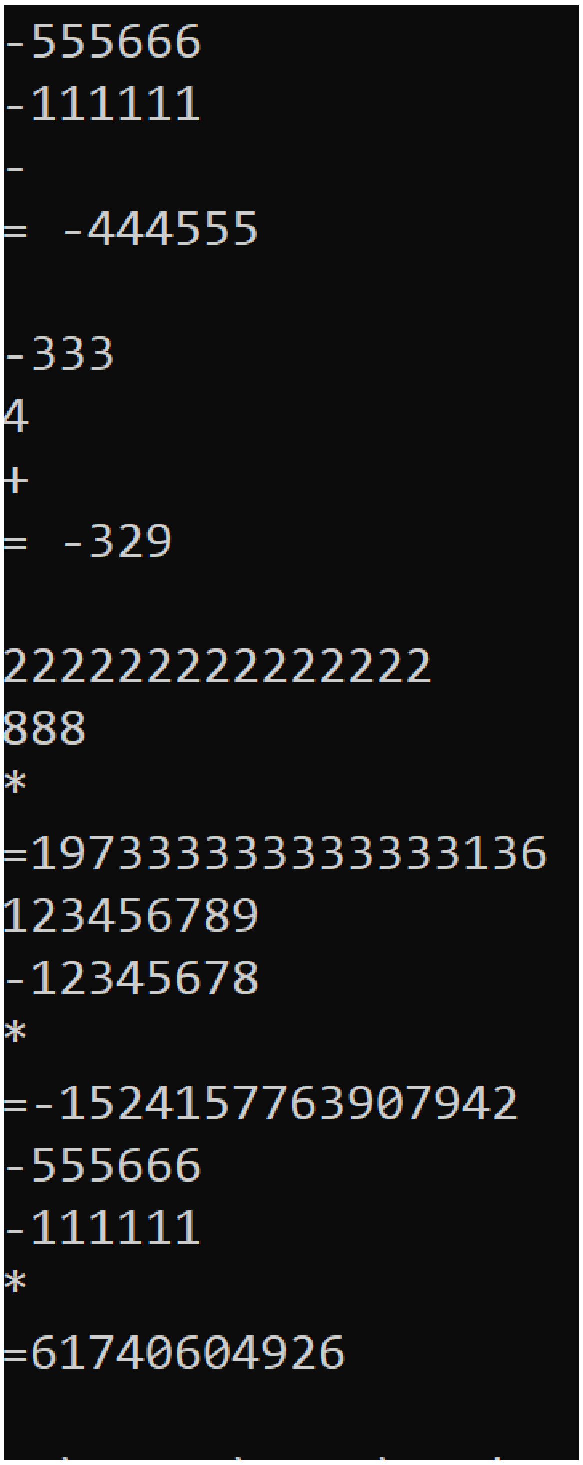
```
9999999999999999999
3333333333333333333
+
=13333333333333333332

2345
-4
+
=2341

123456789
-12345678
-
=135802467

2345
1234
-
=1111

-555666
-111111
```



```
-555666
-111111
-
= -444555

-333
4
+
= -329

222222222222222
888
*
=197333333333333136
123456789
-12345678
*
=-1524157763907942
-555666
-111111
*
=61740604926
```

```
// name: Mariam Tsirekidze

//redID: 823460489

//instr: prof. Tsintsadze

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;


namespace InfiniteNumOperations

{

   class Program

   {

      static void Main(string[] args)

      {

         //read from the file

         string text =File.ReadAllText(@"C:\Users\Asus\Downloads\infint.txt");

         string[] lines = File.ReadAllLines(@"C:\Users\Asus\Downloads\infint.txt");



         InfInt number1 = new InfInt();

         InfInt number2 = new InfInt();



          //calculate until the end of the file

          for (int i = 0; i < lines.Length; ++i)

          {

             if(i%3 == 0)                        //first two strings are numbers
```

```
            {
                number1 = new InfInt(lines[i]);
                Console.WriteLine(lines[i]);
            }
            else if(i%3 == 1)                   //first two strings are numbers
            {
                number2 = new InfInt(lines[i]);
                Console.WriteLine(lines[i]);
            }
            else                                //third item is operator
            {
                if(lines[i] == "+")             //in case of +
                {
                    if (number1.minus == false && number2.minus == false)

                    {
                        Console.WriteLine(lines[i]);
                        Console.WriteLine("=" + number1.Addition(number2)); //if both are positive just add
                        Console.WriteLine();
                    }
                    else if(number1.minus == true && number2.minus == false)
                    {
                        Console.WriteLine(lines[i]);
                        Console.WriteLine("=" + number2.Substraction(number1)); //if the first is negative,
substract first from second
                        Console.WriteLine();
                    }
                    else if(number1.minus == false && number2.minus == true)
                    {
```

```
                Console.WriteLine(lines[i]);
                Console.WriteLine("=" + number1.Substraction(number2)); //if the second is negative, 
substract second from first
               Console.WriteLine();
            }
            else if(number1.minus == true && number2.minus == true)
            {
                Console.WriteLine(lines[i]);
                Console.WriteLine("=-" + number1.Addition(number2)); // if both are negative, add 
them and insert '-' in front
                Console.WriteLine();
            }
        }
        else if(lines[i] == "-")
        {
            if(number1.minus == false && number2.minus == false)     //if both are positive just 
substract second from first
           {
                Console.WriteLine(lines[i]);
                Console.WriteLine("=" + number1.Substraction(number2));
                Console.WriteLine();
            }
            else if(number1.minus == true && number2.minus == false)  //if the first is negative, add 
numbers and insert '-' in front
           {
                Console.WriteLine(lines[i]);
                Console.WriteLine("=-" + number1.Addition(number2));
                Console.WriteLine();
            }
```

```
                        else if(number1.minus == false && number2.minus == true)  //if the second is negative,
just add
                       {
                            Console.WriteLine(lines[i]);
                            Console.WriteLine("=" + number1.Addition(number2));
                            Console.WriteLine();
                        }
                        else if(number1.minus == true && number2.minus == true)  //if both are negative,
substract first from second
                       {
                            Console.WriteLine(lines[i]);
                            Console.WriteLine("=" + number2.Substraction(number1));
                            Console.WriteLine();
                        }
                    }
                   else if (lines[i] == "*")
                   {
                       if (number1.minus == false && number2.minus == false)     //if both are positive just
multiply
                       {
                           Console.WriteLine(lines[i]);
                           Console.Write("=" );
                           number2.Multiplication(number1).ForEach(Console.Write);
                           Console.WriteLine();
                       }
                       else if (number1.minus == true && number2.minus == false) //if first is negative, multiply
and insert '-' in front
                       {
                           Console.WriteLine(lines[i]);
                           Console.Write("=-" );
                           number2.Multiplication(number1).ForEach(Console.Write);

                           Console.WriteLine();

                       }

                       else if (number1.minus == false && number2.minus == true)  //if second is negative,
multiply and insert '-' in front

                       {

                           Console.WriteLine(lines[i]);

                           Console.Write("=-" );

                           number2.Multiplication(number1).ForEach(Console.Write);

                           Console.WriteLine();

                       }

                       else if (number1.minus == true && number2.minus == true) ////if both are negative, just
multiply

                       {

                           Console.WriteLine(lines[i]);

                           Console.Write("=");

                           number2.Multiplication(number1).ForEach(Console.Write);

                           Console.WriteLine();

                       }

                   }


               }
            }








        }

      }

    }
```

```
// name: Mariam Tsirekidze
//redID: 823460489
//instr: prof. Tsintsadze

using System;
using System.Collections.Generic;
using System.Dynamic;
using System.Text;

namespace InfiniteNumOperations
{
  public class InfInt : IComparable<InfInt>
  {
    private List<int> number = new List<int>();     //encapsulate just number
    public bool minus;



    //setter
    public void SetInfInt(String line)
    {
      if (line.StartsWith("-"))          //determine the sign
      {
        this.minus = true;
        line = line.Trim('-');
      }
      else
      {
        this.minus = false;
      }
    int length = line.Length;
    for (int i = 0; i< length; ++i)          //wtite string in the list of ints
    {
        this.number.Add(line[i] - 48);
    }

}

//getter
public List <int>  GetInfInt()
{

    return this.number;

}

//constructor
public InfInt(string line)
{
    if (line.StartsWith("-"))      //determine the sign
    {
        this.minus = true;
        line= line.Trim('-');
    }
    else
    {
        this.minus = false;
    }
    int length = line.Length;

    for (int i = 0; i < length; ++i)  //wtite string in the list of ints

    {

        this.number.Add(line[i] - 48);

    }


}


//default constructor

public InfInt()

{

    this.number.Add(0);

    this.minus = false;

}


public int CompareTo(InfInt other)

{

    int elements1 = this.number.Count;

    int elements2 = other.number.Count;


    if (elements1 == elements2)          //if numbers are the same size

    {

        for(int i = 0; i < elements1; ++i)

        {

            if (this.number[i] != other.number[i])  //compare each digit to each other

            {

                return this.number[i].CompareTo(other.number[i]);

            }
            }
          }
          if(elements1 > elements2)             //if the first is longer return positive
          {
            return 1;
          }
          if(elements2 > elements1)           //if the second is longer return negative
          {
            return -1;
          }
          else                              //else return 0
          {
            return 0;
          }
        }


       public String Addition(InfInt other)
       {
          int i;
          int elements1 = this.number.Count;    //count the number of elements of two numbers
          int elements2 = other.number.Count;
          List<int> sum = new List<int> ();     //create list for the answer sum = number1+number2
          i = elements2;                      //assume that the second number is less then first
          int carry = 0;                      //aasign carry and add to 0
          int add = 0;

          if (this.CompareTo(other)<0)          //if first number is less then second change the value of i to
number of elements in number1
```

```
        {
          i = elements1;
        }
        for(int j = 1; j < i+1; ++j)          // loop until the shorter number ends
        {
            add = this.number[elements1 - j] + other.number[elements2-j]+carry;   // start summing from
the very last digits
            if(add>9)                            //if the sum of diggits are more then 9 we have carry 1
            {
                add = add % 10;
                carry = 1;
            }
            else                               //if the sum is less then or equal to 9 we have carry 0
            {
                carry = 0;
            }
            sum.Insert(0, add);                //insert the number from the top of the list
        }

        for(int j = 1; j < Math.Abs(elements1 - elements2)+1; ++j)      // if we haven't the numbers same
size we should continue summing
        {
            if (CompareTo(other)<0)                                //if first number is shorter we have second number
remaining
            {
                add = other.number[Math.Abs(elements1 - elements2) - j] + carry;
                if (add > 9)
                {
                    add = add % 10;
                    carry = 1;
                }
                else
                {
                    carry = 0;
                }
                sum.Insert(0, add);
            }
            else if (CompareTo(other)>0)                //if second number is shorter we have first
number remaining
            {
                add = this.number[Math.Abs(elements1 - elements2) - j] + carry;
                if (add > 9)
                {
                    add = add % 10;
                    carry = 1;
                }
                else
                {
                    carry = 0;
                }
                sum.Insert(0, add);
            }
        }
        if (carry != 0)         //if addition has ended but we have carry 1 we should insert 1 to the "sum"
list
        {
            sum.Insert(0, 1);
        }
```

```
            String result = String.Join("", sum.ToArray());
            return result;
                        // return the result of addition
        }



        public String Substraction(InfInt other)
        {
            int elements1 = this.number.Count;    //count the number of elements of two numbers
            int elements2 = other.number.Count;
            List<int> sub = new List<int>();     //create list for the answer
            int minuend = 0;
            int difference = 0;
            int carry = 0;
            if (CompareTo(other)<0)     // first number < second, sub =-( number2-number1)
            {

                for(int i = 1; i< elements1+1; ++i )     //until shorter/less number ends
                {

                    minuend = other.number[elements2 - i];
                    if (other.number[elements2-i] - this.number[elements1-i] < 0)     //if the substraction is less
then 0
                    {
                        minuend += 10;
                        carry = 1;
                    }
                    difference = minuend - this.number[elements1 - i];
                    sub.Insert(0, difference);                                //write the difference
                if((elements1 != elements2) && other.number[elements2 - i - 1] > 0 )  //if we have carry, we
should substract from second number
                {
                    other.number[elements2 - i - 1] = other.number[elements2 - i - 1] - carry;
                    carry = 0;
                }
            }

            for(int i = 1; i<(elements2-elements1)+1; ++i)            //continue substracting after shorter/less
number ends
            {
                if (carry == 1)                        //if we still have carry, substract
                {
                    if (other.number[elements2- elements1 - i ] > 0)
                    {
                        other.number[elements2- elements1 - i ] = other.number[elements2 - elements1 - i] -
carry;
                        difference = other.number[elements2 - elements1 - i];
                        sub.Insert(0, difference);
                        carry = 0;
                    }
                }
                else if(carry == 0)            //if we don't have carry just rewrite second number's digits
                {
                    difference = other.number[elements2 - elements1 - i];
                    sub.Insert(0, difference);
                }
            }

            String result = String.Join("", sub.ToArray());    //return answer as a string
```

```
    return result.Insert(0, " -"); ;
}
else if(CompareTo(other) > 0) //same things happen if the first number is more then second
{
    for (int i = 1; i < elements2 + 1; ++i)
    {

        minuend = this.number[elements1 - i];
        if (this.number[elements1 - i] - other.number[elements2 - i] < 0)
        {
            minuend += 10;
            carry = 1;
        }
        difference = minuend - other.number[elements2 - i];
        sub.Insert(0, difference);
        if ((elements1!=elements2) && this.number[elements1 - i - 1] > 0)
        {
            this.number[elements1 - i - 1] = this.number[elements1 - i - 1] - carry;
            carry = 0;
        }
    }

    for (int i = 1; i < (elements1 - elements2) + 1; ++i)
    {
        if (carry == 1)
        {
            if (this.number[elements1 - elements2 - i] > 0)
            {
                this.number[elements1 - elements2 - i] = this.number[elements1 - elements2 - i] - carry;
```

```
                    difference = this.number[elements1 - elements2 - i];
                    sub.Insert(0, difference);
                    carry = 0;
                }
            }
            else if (carry == 0)
            {
                difference = this.number[elements1 - elements2 - i];
                sub.Insert(0, difference);
            }
        }
        String result = String.Join("", sub.ToArray());
        return result;

    }
    else  //if numbers are the same just return 0
    {
        sub.Add(0);
        String result = String.Join("", sub.ToArray());
        return result;
    }
}

public List<int>  Multiplication(InfInt other)
{
    int elements1 = this.number.Count;
    int elements2 = other.number.Count;
    int product;
    int carry = 0;
```

```
        String mulToString;
        InfInt item1 = new InfInt();
        InfInt mul = new InfInt();

        if (CompareTo(other)<=0)        //if first number is less then other
        {
          String[] item = new String [elements1];      //create array of strings
          for (int i = 1; i < elements1+1; ++i)       //until the shorter number ends
          {
            item[i - 1] = "0";                     //write at least 1 element in order to have ability to insert in the
string
            for (int j = 1; j< elements2+1; ++j)    //until the larger number ends
            {
              product = this.number[elements1 - i] * other.number[elements2 - j]+carry;
              if(product>9)                    //if we have carry
              {
                carry = product / 10;
                product = product % 10;

              }
              else
              {
                carry = 0;
              }
              item[i - 1] = item[i - 1].Insert(0, product.ToString());  //save data
            }
            if (carry > 0)                        //after that if we still have carry
            {
              if(carry>9)                    //if carry is 2digit we should add last diggit and then first
              {
                item[i - 1] = item[i - 1].Insert(0, (carry%10).ToString());
                item[i - 1] = item[i - 1].Insert(0, (carry / 10).ToString());
                carry = 0;
              }
              item[i-1] = item[i-1].Insert(0, carry.ToString());
              carry = 0;


            }
          }


          mul = new InfInt(item[0]);            //assign mul as the first number that we have to add
          for(int i = 1; i < elements1; ++i)
          {
            for (int j = 0; j < i; ++j)
            {
              item[i] = item[i].Insert(item[i].Length - 1, "0");
            }
            item1 = new InfInt(item[i].ToString()); //convert the string of number that we have to add in
InfInt
            mulToString = mul.Addition(item1);   //add numbers
            mul = new InfInt(mulToString);      //converts answer in InfInt(in case of another addition)
          }
          mul.GetInfInt().RemoveAt(mul.GetInfInt().Count - 1); //remove unnecessary 0 from the end
          return mul.GetInfInt();                       //return list of integers
        }
        else                        //same things happen if the second number is more than other
```

```
{
    String[] item = new String[elements2];
    for (int i = 1; i < elements2 + 1; ++i)
    {
        item[i - 1] = "0";
        for (int j = 1; j < elements1 + 1; ++j)
        {
            product = this.number[elements1 - j] * other.number[elements2 - i] + carry;
            if (product > 9)
            {
                carry = product / 10;
                product = product % 10;


            }
            else
            {
                carry = 0;
            }
            item[i - 1] = item[i - 1].Insert(0, product.ToString());
        }
        if (carry > 0)
        {
            if (carry > 9)
            {
                item[i - 1] = item[i - 1].Insert(0, (carry % 10).ToString());
                item[i - 1] = item[i - 1].Insert(0, (carry / 10).ToString());
                carry = 0;
            }
            item[i - 1] = item[i - 1].Insert(0, carry.ToString());
```

```
                        carry = 0;

                    }
                }

                mul = new InfInt(item[0]);
                for (int i = 1; i < elements2; ++i)
                {
                    for (int j = 0; j < i; ++j)
                    {
                        item[i] = item[i].Insert(item[i].Length - 1, "0");
                    }
                    item1 = new InfInt(item[i].ToString());
                    mulToString = mul.Addition(item1);
                    mul = new InfInt(mulToString);
                }
                mul.GetInfInt().RemoveAt(mul.GetInfInt().Count - 1);
                return mul.GetInfInt();
            }
        }
    }
}
```