***Project Name :- RSA***

***Team (10)***

*Amir Tareq El Sayed*  ***Section (4)***

*Eslam Nasser Emam* ***Section (3)***

*Ahmed Sherif Morgan*  ***Section (2)***

*Ahmed Saeed Abd el Gowad*  ***Section (1)***

*Zead Mamdouh Ramadan*  ***Section (7)***

* ***Source code of the BigInteger with add function***

public static int[] ADD(int[] arr1, int[] arr2) //O(N)

{

int[] R,R\_NoCarry; // O(1)

int size = arr1.Length;

int result = 0; // O(1)

bool carry\_flag = false; // O(1)

R = new int[size+ 1]; // O(1)

for (int i = size - 1; i >= 0; i--) //O(N \* {O(1)})

{

if (carry\_flag) { result += 1; carry\_flag = false; } // O(1)

result += arr1[i] + arr2[i]; // O(1)

if (result > 9) carry\_flag = true; // O(1)

result = result % 10; // O(1)

R[i + 1] = result; // O(1)

result = 0; // O(1)

}

if (carry\_flag) R[0] = 1; // O(1)

else {

R\_NoCarry = new int[size];

for (int i=size-1;i>=0;i--) //O(N)

{

R\_NoCarry[i] = R[i+1];

}

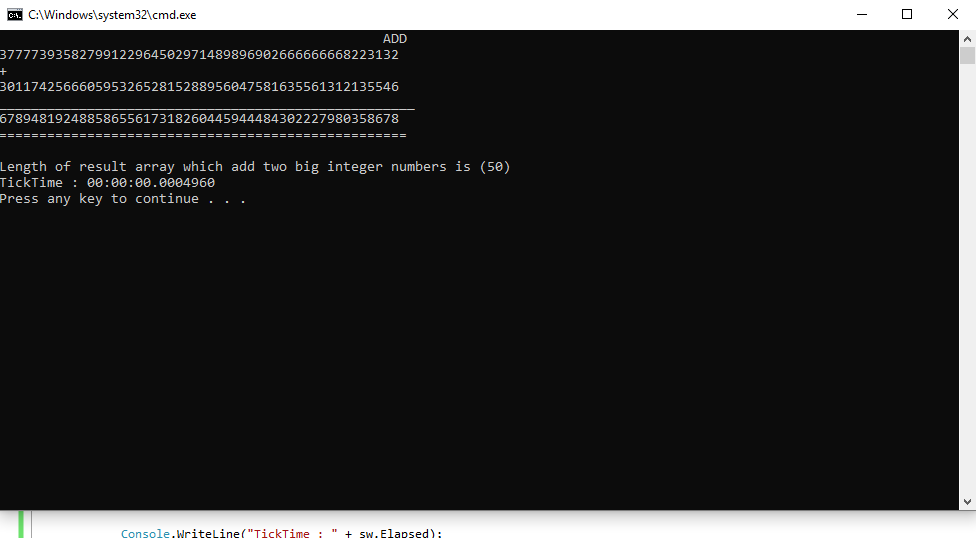
return R\_NoCarry;

}

return R; // O(1)

}

* ***Sample Run for Add function with Execution Time***

******

* ***Source code of the BigInteger with subtraction function***

public static int[] SUB(int[] arr1, int[] arr2) //O(N)

{

int[] R; //O(1)

int size = arr1.Length;

int result = 0; //O(1)

int carry\_Amout = 0; //O(1)

R = new int[size]; //O(1)

for (int i = size - 1; i >= 0; i--) //O(N \* {O(1)})

{

result = arr1[i] - arr2[i] -carry\_Amout; //O(1)

if (result < 0) { result += 10; carry\_Amout = 1; } //O(1)

else carry\_Amout = 0; //O(1)

// r = r % 10;

R[i] = result; //O(1)

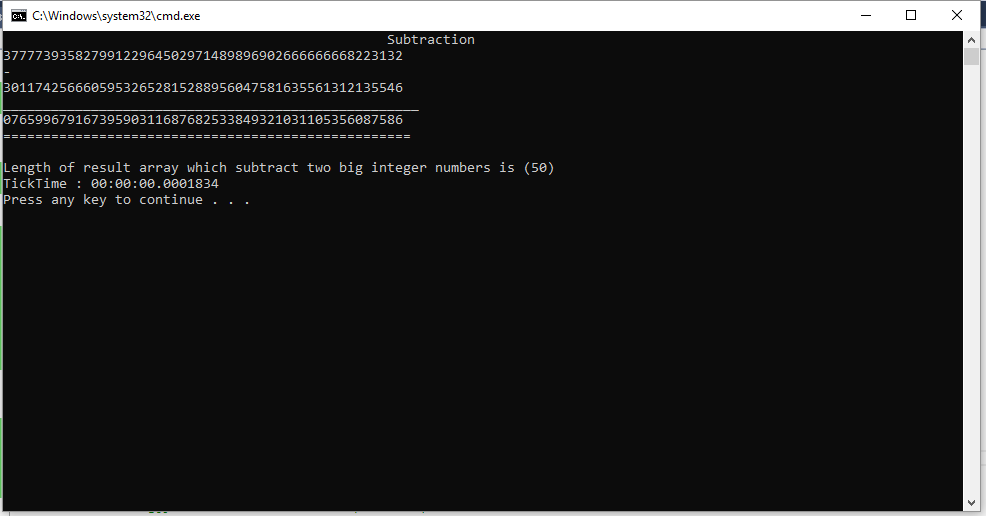
result = 0; //O(1)

}

return R; //O(1)

}

* ***Sample Run for subtraction function with Execution Time***

******

* ***Source code of the BigInteger with multiplication function***

public static int[] Multiply(int[] X, int[] Y) //T(N)=3T(N/2)+O(N) Master Method case 1 F(n)=O(N) g(x)=N^log3 base(2)

//Complexity == O(N^1.59)

{

// 10^n[X1Y1]+[X2Y2]+([X1+X2][Y1+Y2])10^n/2

int[] Failed = { 0 }; //O(1)

int size\_of\_sub\_prob,N; //O(1)

// 10^n[A]+[C]+[B]10^n/2 // Karatsuba Multiplication

int[] A // [X1Y1]

, C // [X2Y2]

, B // Z-(A+C)

, Z, Zx, Zy// [X1+X2][Y1+Y2]

, AC; //O(1)

int a,c,b,z,result; //O(1)

int[] x1, x2

, y1, y2; //O(1)

Make\_Equle(ref X, ref Y); // make Equle length O(N)

int x\_size = Even\_Length(ref X); // check if has even length of Divide equal //O(N)

int y\_size = Even\_Length(ref Y); // ~ //O(N)

N = X.Length; // set N O(1)

if (x\_size == y\_size) size\_of\_sub\_prob = x\_size; //O(1)

else { return Failed; } //O(1)

x1 = new int[size\_of\_sub\_prob]; //O(1)

x2 = new int[size\_of\_sub\_prob]; //O(1)

y1 = new int[size\_of\_sub\_prob]; //O(N)

y2 = new int[size\_of\_sub\_prob]; //O(N)

Divide\_into2Array(ref x1,ref x2,ref X,size\_of\_sub\_prob); //O(N)

Divide\_into2Array(ref y1,ref y2,ref Y,size\_of\_sub\_prob); //O(N)

if (N == 2) //Base Case //O()

{

a = x1[0] \* y1[0];

c = x2[0] \* y2[0];

z = (x1[0] + x2[0]) \* (y1[0] + y2[0]);

b = z - (a + c);

result = (a \* Ten\_power(2)) + c + (b \* Ten\_power(1)); //O(N)

string Res=result.ToString(); //O(1)

return convert\_CharArr\_IntArr(Res.ToCharArray()); //O(N)\*O(1) =O(N)

} // Divide And Conqure

A = Multiply(x1, y1);

C = Multiply(x2, y2);

Make\_Equle(ref x1, ref x2); //O(N)

Zx = ADD(x1, x2); //O(N)

Make\_Equle(ref y1, ref y2); //O(N)

Zy = ADD(y1, y2); //O(N)

Z = Multiply(Zx, Zy);

// B = SUB(Z, ADD(A, C));

Make\_Equle(ref A, ref C); //O(N)

AC = ADD(A, C); //O(N)

Make\_Equle(ref Z, ref AC); //O(N)

B = SUB(Z, AC); //O(N)

// combine

Append\_Zeros(ref A, N); // A 10^N //O(N)

Append\_Zeros(ref B, N / 2); // B 10^N/2 //O(N)

Make\_Equle(ref A, ref C); //O(N)

AC = ADD(A, C); //O(N)

Make\_Equle(ref AC,ref B); //O(N)

return ADD(AC, B); //O(N)

//return Failed;

}

public static void Make\_Equle(ref int[] X, ref int[] Y) //O(N)

{

if (X.Length != Y.Length)

{

if (X.Length > Y.Length)

{

Add\_Zero\_onLeft(ref Y, X.Length - Y.Length);

}

else

{

Add\_Zero\_onLeft(ref X, Y.Length - X.Length);

}

}

}

public static int Even\_Length(ref int[] X) //O(N)

{

int size;

if (X.Length % 2 == 0) { }

else { Add\_Zero\_onLeft(ref X,1); }

size = X.Length / 2;

return size;

}

public static void Divide\_into2Array(ref int[] X1,ref int []X2,ref int [] X ,int size) //O(N)

{

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

{

if (i < size)

{

X1[i] = X[i];

}

else {

X2[i - size] = X[i];

}

}

}

public static void Append\_Zeros(ref int [] arr,int size) //10^n //O(N)

{

int []R=new int [arr.Length+size];

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

{

R[i] = arr[i];

} arr = R;

}

public static void Add\_Zero\_onLeft(ref int[] arr ,int N\_Zeros) //O(N)

{

int[] R = new int[arr.Length+N\_Zeros]; // arr.Length + 1 Default

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

{

R[i+N\_Zeros] = arr[i]; // shift number of zeros

}

arr = R;

}

public static int makeEqualLength(ref string str1, ref string str2)

{

int len1 = str1.Length;

int len2 = str2.Length;

if (len1 < len2)

{

for (int i = 0; i < len2 - len1; i++) //O(N)

str1 = '0' + str1;

return len2;

}

else if (len1 > len2)

{

for (int i = 0; i < len1 - len2; i++) //O(N)

str2 = '0' + str2;

}

return len1; // If len1 >= len2

}

public static int[] convert\_CharArr\_IntArr(char[] ch\_arr) //O(N)

{

int[] int\_arr = new int[ch\_arr.Length]; // O(1)

for (int i = 0; i < int\_arr.Length; i++) // O(N)

{ string s = ch\_arr[i].ToString();

int\_arr[i] = Int32.Parse(s);

}

return int\_arr;

}

public static int Ten\_power(int N) //T(N)=T(N-1)+O(1) // O(N)

{

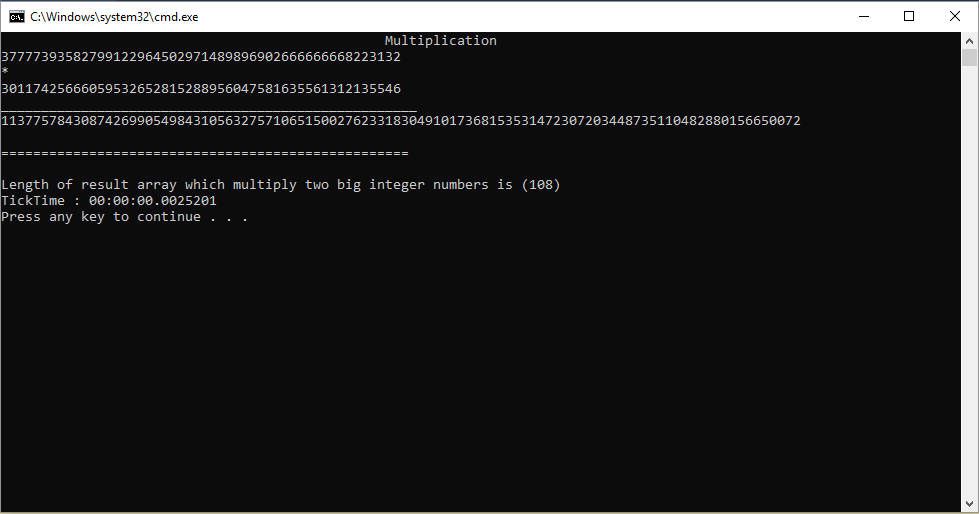
if (N == 0) return 1;

else

return 10\*Ten\_power(N - 1);

}

* ***Sample Run for Multiplication function with Execution Time***

******