***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)

{

Make\_Equle(ref arr1, ref arr2); //O(N)

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

int size = arr1.Length; //O(1)

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]; //O(1)

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

{

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

}

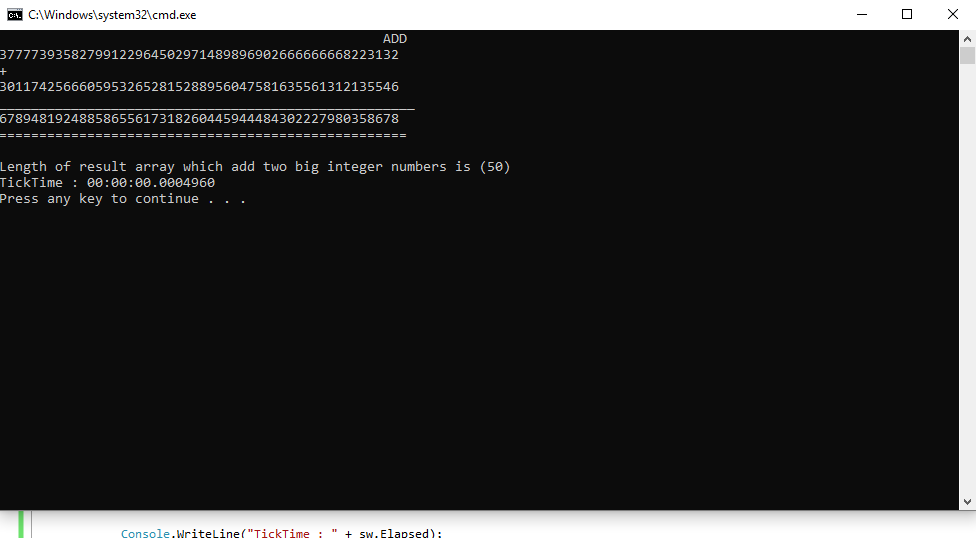
return R\_NoCarry; //O(1)

}

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)

{

Make\_Equle(ref arr1, ref arr2); //O(N)

int[] R; //O(1)

int size = arr1.Length; //O(1)

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)

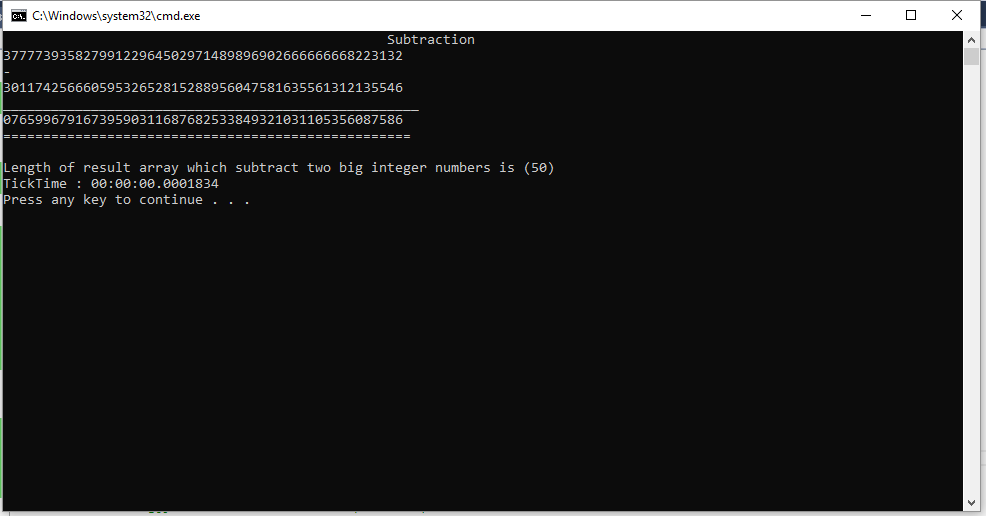
}

Remove\_Zeros\_FromLeft(ref R); //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

, Result; //O(1)

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

int[] x1, x2 //O(1)

, 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]; //O(1)

c = x2[0] \* y2[0]; //O(1)

z = (x1[0] + x2[0]) \* (y1[0] + y2[0]); //O(1)

b = z - (a + c); //O(1)

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

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

return

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

} // Divide And Conqure

A = Multiply(x1, y1); //O(N^1.5)

C = Multiply(x2, y2); //O(N^1.5)

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

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

Z = Multiply(Zx, Zy); //O(N^1.5)

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

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

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

//Zx = Multiply\_Morgan(x1, y2); // X1\*Y2 + X2\*Y1

//Zy = Multiply\_Morgan(x2, y1);

//Make\_Equle(ref Zx, ref Zy);

//Z = ADD(Zx, Zy);

// return ADD(ADD(Append\_Zeros(ref A,A.Length+N),C),Append\_Zeros(ref Z,Z.Length+N/2)); // combine

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

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

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

Result= ADD(AC, B); //O(N)

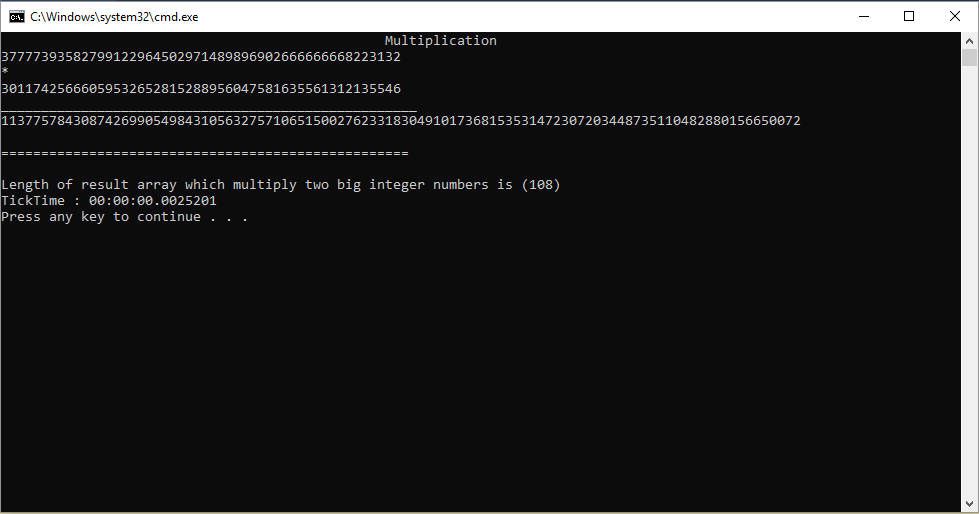
Remove\_Zeros\_FromLeft(ref Result); //O(N)

return Result; //O(1)

//return Failed;

}

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

******

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

//Tuble.item1=Divide | Tuble.item2=Reminder

public static Tuple<int[], int[]> div\_mod(int[] arrA, int[] arrB) //T(N)=T(N)+O(N^1.5) //O(N^1.5)

{

int[] zero = { 0 }, one = { 1 }, mul\_By2 = { 2 }; //O(1)

int[] \_result, \_reminder; //O(1)

Tuple<int[], int[]> Q\_R; //O(1)

if (Compare(ref arrA, ref arrB)) //O(N)

return new Tuple<int[], int[]>(zero, arrA); //O(1)

Q\_R = div\_mod(arrA, Multiply(arrB, mul\_By2)); //O(N^1.5) // Recursive

\_result = Q\_R.Item1; //O(1)

\_result = Multiply(\_result, mul\_By2); //O(N^1.5)

\_reminder = Q\_R.Item2; //O(1)

if (Compare(ref \_reminder, ref arrB)) //O(N)

{

return new Tuple<int[], int[]>(\_result, \_reminder); //O(1)

}

else

{

return new Tuple<int[], int[]>(ADD(\_result, one), SUB(\_reminder, arrB)); //O(N)

}

}

RSA Function

public static int[] RSA(int[] \_base, int[] pow, int[] mod) //T(N)=T(N/2)+O(N^1.5) //O(N^1.5)

//Master Method Case 3 f(x)=O(N^1.5) g(x)=N^(log 1 base 2)

{

//int count = 0; //O(1)

//Console.Write(count++); //O(1)

//Console.Clear(); //O(1)

Tuple<int[], int[]> tuple\_pow, tuple\_mod, tuple\_res; //O(1)

int[] zero = { 0 }, one = { 1 }, two = { 2 }; //O(1)

int[] Result, div\_by2, remind; //O(1)

tuple\_mod = div\_mod(\_base, mod); // Double used //O(N^1.5)

if (pow.Length == 1 && pow[0] == 0) //O(1)

{

return one; //O(1)

}

else if (\_base.Length == 1 && \_base[0] == 0) //O(1)

{

return zero; //O(1)

}

else if (pow.Length == 1 && pow[0] == 1) //O(1)

{

// tuble = div\_mod(\_base, mod);

return tuple\_mod.Item2; //O(1)

}

else

{

tuple\_pow = div\_mod(pow, two); //O(N^1.5)

div\_by2 = tuple\_pow.Item1; //O(1)

remind = tuple\_pow.Item2; //O(1)

if (remind.Length == 1 && remind[0] == 0) //O(1)

{

Result = RSA(\_base, div\_by2, mod);

Result = Multiply(Result, Result); //O(N^1.5)

}

else

{

Result = RSA(\_base, div\_by2, mod);

Result = Multiply(Result, Result); //O(N^1.5)

remind = tuple\_mod.Item2; //O(1)

Result = Multiply(Result, remind); //O(N^1.5)

}

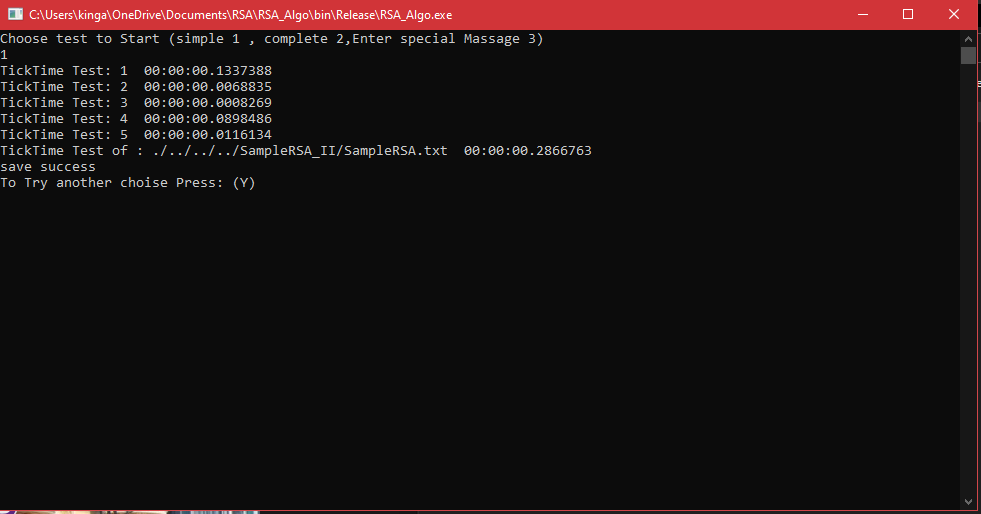
}

tuple\_res = div\_mod(Result, mod); //O(N^1.5)

return tuple\_res.Item2; //O(1)

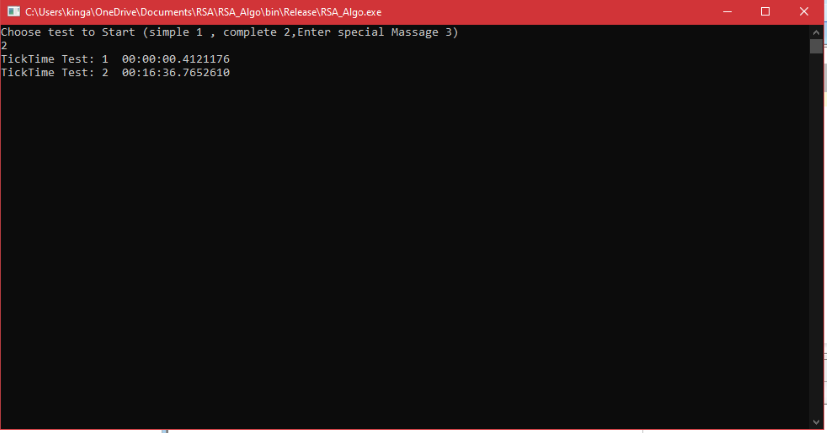
}

* ***Sample run for simple test cases && Execution Time***

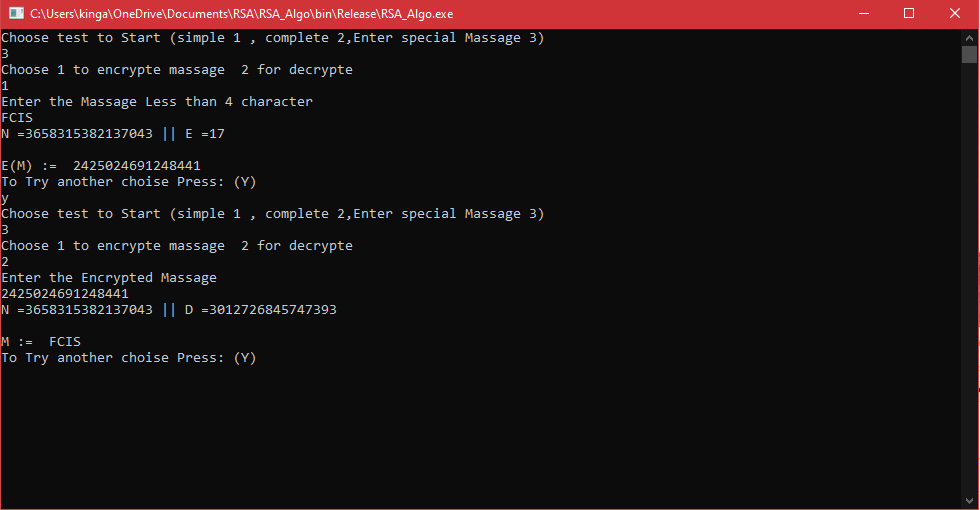
******

* ***Sample run for complete test cases && Execution Time***

***Except ( last two cases can’t be computed because the low performance of the device )***

******

* ***Sample run for Bonus 1***

******

* ***Source code of functions which help Project code***

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

{

if (X.Length != Y.Length) //O(1)

{

if (X.Length > Y.Length) //O(1)

{

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

}

else

{

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

}

}

}

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

{

int size; //O(1)

if (X.Length % 2 == 0) { } //O(1)

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

size = X.Length / 2; //O(1)

return size; //O(1)

}

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++) //O(N)

{

if (i < size) //O(1)

{

X1[i] = X[i]; //O(1)

}

else

{

X2[i - size] = X[i]; //O(1)

}

}

}

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

{

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

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

{

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

}

// return R;

arr = R; //O(1)

}

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++) //O(N)

{

R[i + N\_Zeros] = arr[i]; // shift number of zeros //O(1)

}

arr = R; //O(1)

}

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

{

int len1 = str1.Length; //O(1)

int len2 = str2.Length; //O(1)

if (len1 < len2) //O(1)

{

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

str1 = '0' + str1; //O(1)

return len2; //O(1)

}

else if (len1 > len2) //O(1)

{

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

str2 = '0' + str2; //O(1)

}

return len1; // If len1 >= len2 //O(1)

}

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

{

if (N == 0) return 1; //O(1)

else

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

}

public static void Remove\_Zeros\_FromLeft(ref int[] int\_arr) //O(N)

{

int c = 0; //O(1)

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

{

if (int\_arr[i] == 0) //O(1)

{

c++; //O(1)

continue; //O(1)

}

else

{

break; //O(1)

}

}

int[] v; //O(1)

if (int\_arr.Length == c) //O(1)

{

v = new int[1]; //O(1)

int\_arr = v; //O(1)

return ; //O(1)

}

v= new int[int\_arr.Length - c]; //O(1)

int t = 0; //O(1)

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

{

v[t] = int\_arr[i]; //O(1)

t++; //O(1)

}

int\_arr = v; //O(1)

return; //O(1)

}

public static bool Compare(ref int[] arr1, ref int[] arr2)//retun true when arr1<arr2 //O(N)

{

int size=arr1.Length; //O(1)

if (arr1.Length < arr2.Length) //O(1)

return true; //O(1)

else if (arr2.Length < arr1.Length) //O(1)

return false; //O(1)

else

{

for (int i = 0; i < size; i++) //O(N)

{

if (arr1[i] == arr2[i]) //O(1)

continue; //O(1)

else {

if (arr1[i] < arr2[i]) //O(1)

return true; //O(1)

else

return false; //O(1)

}

}

}

return false; //O(1)

}

//Convert\_string\_To\_intArr

public static int[] AsciiCode(string Message) // N \* O(1) = O(N)

{

//string s;

int j = 0,ascii,mod1,mod2; //O(1)

char[] \_m = Message.ToCharArray(); //O(N)

int[] arr = new int[\_m.Length \* 3]; //O(1)

for (int i = 0; i < arr.Length; i+=3) //O(N)

{

//s= \_m[j].ToString();

ascii = (int)\_m[j]; //O(1)

//ascii = int.Parse(s);

if (ascii < 100) //O(1)

{

mod1 = ascii % 10; //O(1)

ascii = ascii / 10; //O(1)

arr[i] = 0; //O(1)

arr[i + 1] = ascii; //O(1)

arr[i + 2] = mod1; //O(1)

}

//O(1)

else

{

mod1 = ascii % 10; //O(1)

ascii = ascii / 10; //O(1)

mod2 =ascii% 10; //O(1)

ascii = ascii / 10; //O(1)

arr[i] = ascii; //O(1)

arr[i + 1] = mod2; //O(1)

arr[i + 2] = mod1; //O(1)

}

j++; //O(1)

}

return arr; //O(1)

}

public static string convert\_Ascii\_To\_String(int[] Arr) //O(N)

{

string Massage = ""; //O(1)

char ch; //O(1)

int ch\_ascii,rem; //O(1)

//int[] three = { 3 }, rem;

rem = (Arr.Length % 3); //O(1)

if (rem != 0) //O(1)

{

Add\_Zero\_onLeft(ref Arr,3-rem); //O(N)

}

for (int i = 0; i < Arr.Length; i += 3) //O(N)

{

ch\_ascii = (Arr[i] \* 100) + (Arr[i + 1] \* 10) + Arr[i + 2]; //O(1)

ch = (char)ch\_ascii; //O(1)

Massage += ch; //O(1)

}

return Massage; //O(1)

}