دانشکده مهندسی کامپیوتر

ساختمان داده

تمرین ۳*

زهرا حسینی
سهراب نمازی
سید صالح اعتمادی

نیم‌سال اول ۰۰-۹۹

ایمیل/تیمز	hosseini_zahra@comp.iust.ac.ir sohrab_namazi@comp.iust.ac.ir
نام شاخه	fb_A3
نام پروژه/پوشه/پول ریکوست	A3
مهلت تحویل	۹۹/۷/۲۶

*تشکر ویژه از خانم مریم سادات هاشمی که در نیم‌سال اول سال تحصیلی ۹۷-۹۸ نسخه اول این مجموعه تمرین‌ها را تهیه فرمودند. همچنین از اساتید حل‌تمرین نیم‌سال اول سال تحصیلی ۹۹-۹۸ سارا کدیری، محمد مهدی عبداللهپور، مهدی مقدمی، مهسا قادران، علیرضا مرادی، پریسا یل‌سوار، غزاله محمودی و محمدجواد میرشکاری که مستند این مجموعه تمرین‌ها را بهبود بخشیدند، متشکرم.

توضیحات کلی تمرین

۱. ابتدا مانند تمرین های قبل، یک پروژه به نام A3 بسازید.

۲. کلاس هر سوال را به پروژهی خود اضافه کنید و در قسمت مربوطه کد خود را بنویسید. هر کلاس شامل دو متد اصلی است:

- متد اول: تابع Solve است که شما باید الگوریتم خود را برای حل سوال در این متد پیاده سازی کنید.
- متد دوم: تابع Process است که مانند تمرین های قبلی در TestCommon پیاده سازی شده است. بنابراین با خیال راحت سوال را حل کنید و نگران تابع Process نباشید! زیرا تمامی پیاده سازی ها برای شما انجام شده است و نیازی نیست که شما کدی برای آن بزنید.

۳. اگر برای حل سوالی نیاز به تابع های کمکی دارید؛ می توانید درکلاس مربوط به همان سوال تابع تان را اضافه کنید.

اکنون که پیاده سازی شما به پایان رسیده است، نوبت به تست برنامه می رسد. مراحل زیر را انجام دهید.

۱. یک UnitTest برای پروژهی خود بسازید.

۲. فولدر TestData که در ضمیمه همین فایل قرار دارد را به پروژهی تست خود اضافه کنید.

۳. فایل GradedTests.cs را به پروژهی تستی که ساخته اید اضافه کنید.

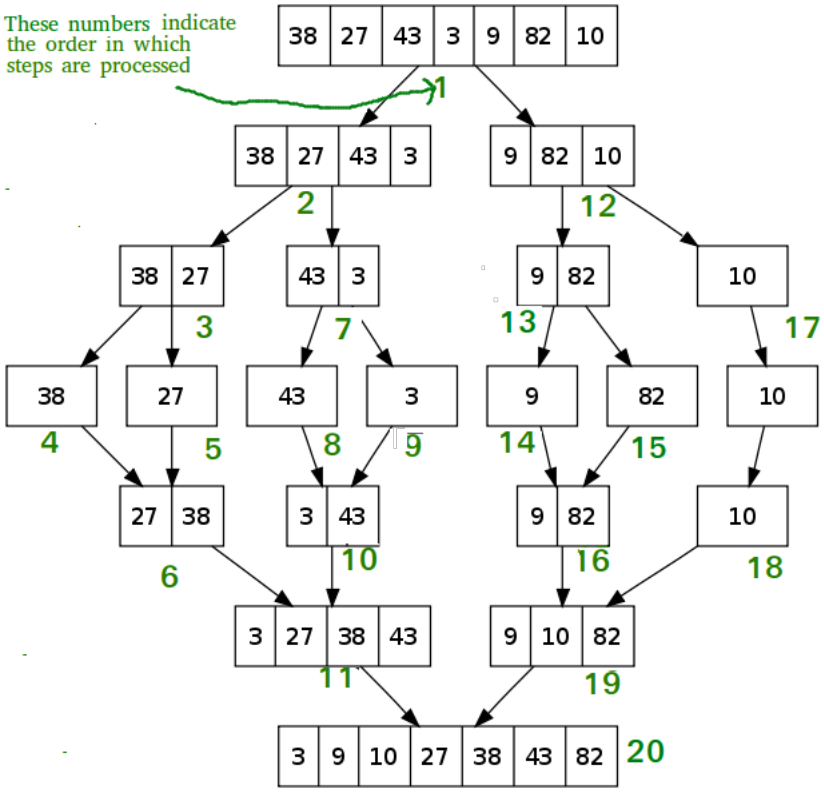
توجه:
برای اینکه تست شما از بهینه سازی کامپایلر دات نت حداکثر بهره را ببرد زمان تست ها را روی بیلد Release امتحان کنید، درغیر اینصورت ممکن است تست های شما در زمان داده شده پاس نشوند.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using TestCommon;

namespace A3.Tests
{
    [TestClass()]
    public class GradedTests
    {
        [TestMethod(), Timeout(400)]
        public void SolveTest_Q1MergeSort()
        {
            RunTest(new Q1MergeSort("TD1"));
        }

        [TestMethod(), Timeout(100)]
        public void SolveTest_Q2Fibonacci()
        {
            RunTest(new Q2FibonacciFast("TD2"));
        }

        [TestMethod(), Timeout(400)]
        public void SolveTest_Q3FibonacciLastDigit()
        {
            RunTest(new Q3FibonacciLastDigit("TD3"));
        }

        [TestMethod(), Timeout(100)]
        public void SolveTest_Q4GCD()
        {
            RunTest(new Q4GCD("TD4"));
        }

        [TestMethod(), Timeout(100)]
        public void SolveTest_Q5LCM()
        {
            RunTest(new Q5LCM("TD5"));
        }

        [TestMethod(), Timeout(100)]
        public void SolveTest_Q6FibonacciMod()
        {
            RunTest(new Q6FibonacciMod("TD6"));
        }

        [TestMethod(), Timeout(100)]
        public void SolveTest_Q7FibonacciSum()
        {
            RunTest(new Q7FibonacciSum("TD7"));
        }

        [TestMethod(), Timeout(100)]
        public void SolveTest_Q8FibonacciPartialSum()
        {
            RunTest(new Q8FibonacciPartialSum("TD8"));
        }

        [TestMethod(), Timeout(100)]
```

```
        public void SolveTest_Q9FibonacciSumSquares()
        {
            RunTest(new Q9FibonacciSumSquares("TD9"));
        }

        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("A3", p.Process, p.TestDataName, p.Verifier,
                VerifyResultWithoutOrder: p.VerifyResultWithoutOrder,
                excludedTestCases: p.ExcludedTestCases);
        }
    }
}
```

۱ Merge Sort

در این تمرین شما باید الگوریتم $MergeSort$ را پیاده سازی کنید. یک مثال از $MergeSort$ در تصویر زیر نمایش داده شده است. روش این الگوریتم به این صورت است که در هر مرحله آرایه ورودی را نصف کرده و روی هر دو قسمت به صورت بازگشتی خود را صدا می زند و پس از بازگشت دو قسمت را با هم $Merge$ می کند.



- به دلیل پیچیدگی بیشتر این سوال نسبت به بقیه سوال ها توصیه می شود ابتدا به بقیه سوالات پاسخ دهید.

- محدودیت زمانی : ۴۰۰ میلی ثانیه

```
using System;
using System.Collections.Generic;
using System.Text;
using TestCommon;

namespace A3
{
    public class Q1MergeSort : Processor
    {
        public Q1MergeSort(string testDataName) : base(testDataName) { }

        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long, long[], long[]>)Solve);

        public long[] Solve(long n, long[] a)
        {
```

```
            throw new NotImplementedException();
        }
    }
}
```

۲ Fibonacci Number

در این تمرین شما باید الگوریتمی بنویسید که با گرفتن عدد صحیح n از ورودی، nامین عدد فیبوناچی را پیدا کند.
تعریف دنباله ی اعداد فیبوناچی به صورت زیر می باشد:

$$Fib(0) = 0, Fib(1) = 1, Fib(i) = Fib(i-1) + Fib(i-2), i >= 2$$

- محدودیت زمانی : ۱۰۰ میلی ثانیه

```
using System;
using TestCommon;

namespace A3
{
    public class Q2FibonacciFast : Processor
    {
        public Q2FibonacciFast(string testDataName) : base(testDataName) { }

        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long, long>)Solve);

        public long Solve(long n)
        {
            throw new NotImplementedException();
        }
    }
}
```

۳ Last Didit of a Large Fibonnaci Number

هدف شما در این تمرین پیدا کردن آخرین رقم nامین عدد فیبوناتچی است. به یاد بیاورید که اعداد فیبوناچی سریعا رشد می کنند. بنابراین باید الگوریتم شما کارآمد باشد.

- محدودیت زمانی : ۴۰۰ میلی ثانیه

```
using System;
using TestCommon;

namespace A3
{
    public class Q3FibonacciLastDigit : Processor
```

```
    {
        public Q3FibonacciLastDigit(string testDataName) : base(testDataName) { }

        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long, long>)Solve);

        public long Solve(long n)
        {
            throw new NotImplementedException();
        }
    }
}
```

۴ Greatest Common Divisor

بزرگترین مقسوم علیه مشترک دو عدد صحیح غیر منفی a و b (که هیچکدام صفر نیستند) برابر است با بزرگترین عدد صحیح مانند d که بر هر دو عدد a و b تقسیم می شود. در این تمرین، الگوریتم اقلیدس را برای محاسبه بزرگترین مقسوم علیه مشترک اجرا کنید.

- محدودیت زمانی : ۱۰۰ میلی ثانیه

```
using System;
using TestCommon;

namespace A3
{
    public class Q4GCD : Processor
    {
        public Q4GCD(string testDataName) : base(testDataName) { }

        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long, long, long>)Solve);

        public long Solve(long a, long b)
        {
            throw new NotImplementedException();
        }
    }
}
```

۵ Least Common Multiple

کوچکترین مضرب مشترک دو عدد صحیح مثبت a و b، حداقل عدد صحیح مثبت m است که توسط a و b قابل تقسیم است. الگوریتمی بنویسید که کوچکترین مضرب مشترک دو عدد صحیح که از ورودی می گیرد را محاسبه کند.

- این سوال امتیازی می باشد.

- محدودیت زمانی : ۱۰۰ میلی ثانیه

```
using System;
using TestCommon;

namespace A3
{
    public class Q5LCM : Processor
    {
        public Q5LCM(string testDataName) : base(testDataName) { }

        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long, long, long>)Solve);

        public long Solve(long a, long b)
        {
            throw new NotImplementedException();
        }

    }
}
```

۶ Fibonacci Number Again

در این تمرین، هدف شما این است که باقی مانده ی $Fib(n)$ بر m را برای مقادیر خیلی بزرگ n مثل 10^{18} را محاسبه کنید. برای چنین مقادیر بزرگی از n، اگر شما یک حلقه برای محاسبه ی عدد فیبوناتچی n استفاده کنید و سپس باقی مانده ی آن را بر m حساب کنید، زمان اجرای الگوریتم شما بیشتر از یک ثانیه خواهد بود. بنابراین باید از روش دیگری استفاده کرد. به شکل زیر دقت کنید.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F_i	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610
$F_i \bmod 2$	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0
$F_i \bmod 3$	0	1	1	2	0	2	2	1	0	1	1	2	0	2	2	1

همانطور که می بینید دنباله ی باقی مانده ها هم برای $m = 2$ و هم برای $m = 3$ متناوب است.به طور کلی این درست است که برای هر عدد صحیح $m \geq ۲$، دنباله $Fib(n) \bmod m$ متناوب است. اثبات می شود که تناوب همیشه با ۰۱ شروع می شود و به عنوان تناوب پیزانو شناخته می شود. پس شما با دانستن این نکته می توانید مسئله را حل کنید.

- محدودیت زمانی : ۱۰۰ میلی ثانیه

```
using System;
using TestCommon;

namespace A3
{
    public class Q6FibonacciMod : Processor
    {
        public Q6FibonacciMod(string testDataName) : base(testDataName) { }
```

```
        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long, long, long>)Solve);

        public long Solve(long a, long b)
        {
            throw new NotImplementedException();
        }
    }
}
```

۷ Last Digit of the Sum of Fibonacci Numbers

الگوریتمی بنویسید که اخرین رقم مجموع $fib(0) + fib(1) + fib(2) + ... + fib(n)$ را محاسبه کند.

- محدودیت زمانی : ۱۰۰ میلی ثانیه

```
using System;
using TestCommon;

namespace A3
{
    public class Q7FibonacciSum : Processor
    {
        public Q7FibonacciSum(string testDataName) : base(testDataName) { }

        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long, long>)Solve);

        public long Solve(long n)
        {
            throw new NotImplementedException();
        }
    }
}
```

۸ Last Digit of the Sum of Fibonacci Numbers Again

دو عدد صحیح غیر منفی m و n ، که در آن $n \geq m$ را از ورودی بگیرید. آخرین رقم مجموع عبارت زیر را پیدا کنید. $Fib(m) + Fib(m + 1) + ... + Fib(n)$

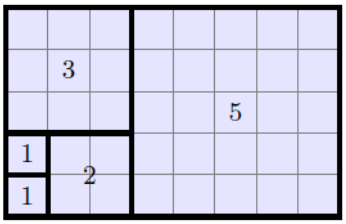
- محدودیت زمانی : ۱۰۰ میلی ثانیه

```
using System;
using TestCommon;

namespace A3
```

```
{
    public class Q8FibonacciPartialSum : Processor
    {
        public Q8FibonacciPartialSum(string testDataName) : base(testDataName) { }

        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long, long, long>)Solve);

        public long Solve(long a, long b)
        {
            throw new NotImplementedException();
        }
    }
}
```

۹ Last Digit of the Sum of Squares of Fibonacci Numbers

آخرین رقم $Fib(0)^2 + Fib(1)^2 + ... + Fib(n)^2$ را محاسبه کنید.
از آنجایی که n در این سوال می تواند خیلی بزرگ باشد؛ یک فرمول برای محاسبه ی عبارت بالا بدست بیاورید.
به شکل زیر دقت کنید.



این شکل نشان می دهد که مجموع $Fib(0)^2 + Fib(1)^2 + Fib(2)^2 + Fib(3)^2 + Fib(4)^2 + Fib(5)^2$
برابر است با مساحت مستطیلی با عرض $Fib(5)$ و طول $Fib(4) + Fib(5)$

- محدودیت زمانی : ۱۰۰ میلی ثانیه

```
using System;
using TestCommon;

namespace A3
{
    public class Q9FibonacciSumSquares : Processor
    {
        public Q9FibonacciSumSquares(string testDataName) : base(testDataName) { }

        public override string Process(string inStr) =>
            TestTools.Process(inStr, (Func<long, long>)Solve);

        public long Solve(long n)
        {
            throw new NotImplementedException();
        }
```

```
    }
}
```