تمرین ۳ درس ساختمان داده

سید صالح اعتمادی
مریم سادات هاشمی

دانشگاه علم و صنعت ۹۸_۹۷

لطفا به نکات زیر توجه کنید:

- مهلت ارسال این تمرین شنبه ۲۱ مهر ماه ساعت ۱۱:۵۹ ب.ظ است.
- این تمرین شامل سوال های برنامه نویسی می باشد، بنابراین توجه کنید که حتماً موارد خواسته شده را رعایت کنید. نام تابع ها و تست ها باید همگی مطابق آنچه که خواسته شده است؛ باشد.
- نام شاخه، پوشه و پول ریکوست همگی دقیقا "A۳" باشد.
- در صورتی که به اطلاعات بیشتری نیاز دارید می توانید با ایدی تلگرام @maryam_sadat_hashemi در ارتباط باشید.
- اگر در حل تمرین شماره ی ۲ مشکلی داشته اید،لطفا به `https://calendly.com/hashemi-maryam-sadat` مراجعه کنید و زمانی را برای رفع اشکال تنظیم کنید.

موفق باشید.

توضیحات کلی تمرین

تمرین این هفته ی شما، ۸ سوال دارد که باید به همه ی این سوال ها پاسخ دهید.
ابتدا مانند تمرین های قبل، یک پروژه به نام A۳ بسازید. شما باید برای هر سوال یک تابع بسازید و الگوریتم خود را داخل آن تابع پیاده سازی کنید. دقت کنید که در هر بخش توضیح دادیم که اسم تابع ها را چگونه انتخاب کنید.
بعد از اینکه الگوریتم خود را در تابع مورد نظر پیاده سازی کردید؛ برای بررسی درستی الگوریتم، شما باید الگوریتم خود را تست کنید. بنابراین شما نیاز دارید یک Unit Test برای پروژه ی خود بسازید. سپس باید فولدر TestData که در ضمیمه همین فایل قرار دارد را به پروژه ی تست خود اضافه کنید. داخل فولدر TestData هشت فولدر دیگر قرار دارد که در هر کدام testcase های هر سوال قرار داده شده است. برای مثال testcase های سوال یک در فولدرTD۱ می باشد.
بعد از اینکه فولدر TestData را به پروژه ی خود اضافه کردید؛ باید برای هر الگوریتم خود یک TestMethod نیز بنویسید. (دقت کنید که در توضحات هر سوال نام TestMethod را هم مشخص کردیم و TestMethod های شما هم دقیقا باید به همین اسامی باشند.) برای نوشتن تست هم از TestCommon (مثل تمرین های قبل) باید استفاده کنید. دقت کنید که تغییر کوچکی در TestCommon برای این تمرین انجام شده است. بنابراین شما باید ابتدا نسخه ی جدید این کلاس را با دستورPull git دریافت کنید .
به شکل زیر دقت کنید.

```
public void Graded_FibonacciTest()
{
   TestCommon.TestTools.RunLocalTest(Program.ProcessFibonacci, "TD1");
}
```

بر خلاف تمرین قبل متد RunLocalTest دو ورودی می گیرد:
ورودی اول: یک تابع است که ورودی و خروجی این تابع از نوع string است.بنابراین شما باید یک تابع process هم برای هر سوال در پروژه ی اصلی خود داشته باشید که به عنوان ورودی به متد RunLocalTest بدهید. هدف از تابع process این است که داده ی مورد نظر را از ورودی می گیرد و سپس پردازش لازم روی دیتای ورودی انجام شود تا ورودی مطابق با نوع ورودی تابع الگوریتم شما باشد. برای مثال اگر تابع الگوریتم شما داده را به صورت int دریافت می کند شما باید داده را به صورت int در بیاورید تا بتوانید داده را به تابع الگوریتم خود بدهید. و در نهایت هم خروجی تابع process همان خروجی تابع الگوریتم شما خواهد بود. دقت کنید که در توضحات هر سوال، نام تابع Process را هم مشخص کردیم و تابع های شما هم دقیقا باید به همین اسامی باشند.(برای چگونگی ساختن تابع process به توضیحات تمرین قبل خود مراجعه کنید یا از delegate استفاده کنید.)

```
public static string Process(string inStr, Func<long, long> longProcessor)
{
    long n = long.Parse(inStr);
    return longProcessor(n).ToString();
}
```

```
public static string ProcessFibonacci(string inStr) =>
    Process(inStr, Fibonacci);
```

ورودی دوم : نام فولدری است که testcase های مربوط به آن سوال قرار دارد مثلا برای سوال یک این ورودی TD۱ خواهد بود و برای سوال دو TD۲ و الی آخر.
اکنون شما موفق شدید تابع الگوریتم و process و TestMethod مربوط به سوال را پیاده سازی کنید و باید الگوریتم خود را تست کنید.

۱ Fibonacci Number

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

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

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

اسامی توابع شما در این تمرین باید به صورت زیر باشد.

- Algorithm Function : Fibonacci
- Process Function : ProcessFibonacci
- Test Function : Graded_FibonacciTest

```
public static long Fibonacci(long n)


public static string ProcessFibonacci(string inStr) =>
    Process(inStr, Fibonacci);


public void Graded_FibonacciTest()
{
   TestCommon.TestTools.RunLocalTest(Program.ProcessFibonacci, "TD1");
}
```

۲ 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 : **۱۰۰۰ میلی ثانیه**

اسامی توابع شما در این تمرین باید به صورت زیر باشد.

- Algorithm Function : Fibonacci_LastDigit
- Process Function : ProcessFibonacci_LastDigit
- Test Function : Graded_FibonacciLastDigitTest

```
public static long Fibonacci_LastDigit(long n)
```

```
public static string ProcessFibonacci_LastDigit(string inStr) =>
    Process(inStr, Fibonacci_LastDigit);
```

```
public void Graded_FibonacciLastDigitTest()
{
    TestCommon.TestTools.RunLocalTest(Program.ProcessFibonacci_LastDigit, "TD2");
}
```

۳ Greatest Common Divisor

بزرگترین مقسوم علیه مشترک دو عدد صحیح غیر منفی a و b (که هر دو برابر ۰ 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 : **۱۰۰۰ میلی ثانیه**

اسامی توابع شما در این تمرین باید به صورت زیر باشد.

- Algorithm Function : GCD
- Process Function : ProcessGCD
- Test Function : Graded_GCDTest

```
public static long GCD(long a, long b)
```

```
public static string ProcessGCD(string inStr) =>
    Process(inStr, GCD);
```

```
public void Graded_GCDTest()
{
    TestCommon.TestTools.RunLocalTest(Program.ProcessGCD, "TD3");
}
```

۴ 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.

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

اسامی توابع شما در این تمرین باید به صورت زیر باشد.

- Algorithm Function : LCM
- Process Function : ProcessLCM
- Test Function : Graded_LCMTest

```
public static long LCM(long a, long b)
```

```
public static string ProcessLCM(string inStr) =>
    Process(inStr, LCM);
```

```
public void Graded_LCMTest()
{
    TestCommon.TestTools.RunLocalTest(Program.ProcessLCM, "TD4");
}
```

۵ 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 : **۱۰۰۰ میلی ثانیه**

اسامی توابع شما در این تمرین باید به صورت زیر باشد.

- Algorithm Function : Fibonacci_Mod
- Process Function : ProcessFibonacci_Mod
- Test Function : Graded_FibonacciModTest

```
public static long Fibonacci_Mod(long n, long m)
```

```
public static string ProcessFibonacci_Mod(string inStr) =>
    Process(inStr, Fibonacci_Mod);
```

```
public void Graded_FibonacciModTest()
{
    TestCommon.TestTools.RunLocalTest(Program.ProcessFibonacci_Mod, "TD5");
}
```

۶ 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 : **۱۰۰۰ میلی ثانیه**

اسامی توابع شما در این تمرین باید به صورت زیر باشد.

- Algorithm Function : Fibonacci_Sum
- Process Function : ProcessFibonacci_Sum
- Test Function : Graded_FibonacciSumTest

```
public static long Fibonacci_Sum(long n)
```

```
public static string ProcessFibonacci_Sum(string inStr) =>
    Process(inStr, Fibonacci_Sum);
```

```
public void Graded_FibonacciSumTest()
{
    TestCommon.TestTools.RunLocalTest(Program.ProcessFibonacci_Sum, "TD6");
}
```

۷ Digit of the Sum of Fibonacci Numbers Again Last

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 : **۱۰۰۰ میلی ثانیه**

اسامی توابع شما در این تمرین باید به صورت زیر باشد.

- Algorithm Function : Fibonacci_Partial_Sum
- Process Function : ProcessFibonacci_Partial_Sum
- Test Function : Graded_FibonacciPartialSumTest

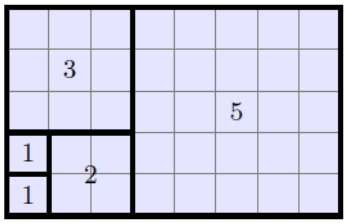
```
public static long Fibonacci_Partial_Sum(long n, long m)
```

```
public static string ProcessFibonacci_Partial_Sum(string inStr) =>
    Process(inStr, Fibonacci_Partial_Sum);
```

```
public void Graded_FibonacciPartialSumTest()
{
    TestCommon.TestTools.RunLocalTest(Program.ProcessFibonacci_Partial_Sum, "TD7");
}
```

۸ 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 $Fib(0)^2+Fib(1)^2+Fib(2)^2+Fib(3)^2+Fib(4)^2$ برابر است با مساحت مستطیلی با عرض $Fib(5)$ و طول $Fib(4) + Fib(5)$ $Fib(5)^2$

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

اسامی توابع شما در این تمرین باید به صورت زیر باشد.

- Algorithm Function : Fibonacci_Sum_Squares
- Process Function :ProcessFibonacci_Sum_Squares
- Test Function : Graded_FibonacciSumSquaresTest

```
public static long Fibonacci_Sum_Squares(long n)
```

```
public static string ProcessFibonacci_Sum_Squares(string inStr) =>
    Process(inStr, Fibonacci_Sum_Squares);
```

```
public void Graded_FibonacciSumSquaresTest()
{
    TestCommon.TestTools.RunLocalTest(Program.ProcessFibonacci_Sum_Squares, "TD8");
}
```