دانشکده مهندسی کامپیوتر

ساختمان داده

تمرین ۷*

مبین داریوش همدانی
بابک بهکام کیا
سید صالح اعتمادی

نیم‌سال اول ۱۴۰۱-۱۴۰۰

ایمیل/تیمز	m_dariushhamedani@comp.iust.ac.ir babak_behkamkia@comp.iust.ac.ir
نام شاخه	fb_A7
نام پروژه/پوشه/پول ریکوست	A7
مهلت تحویل	۱۴۰۰/۸/۲۹

*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 A7 بسازید. همچنین پروژه تست متناظر آن را ساخته و مطابق راهنمای تمرین یک فایل‌ها را در پوشه متناظر اضافه کرده و تنظیمات مربوط به کپی کردن TestData به پوشه خروجی را در تنظیمات پروژه تست قرار دهید. دقت کنید که پروژه TestCommon فقط یکبار باید در ریشه گیت موجود باشد و نباید در هر تمرین مجدد کپی شود. برای روش ارجاع به این پروژه به تمرین شماره یک مراجعه کنید.

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 در کلاس مربوط به همان سوال تابع تان را 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 A7.Tests
{
    [DeploymentItem("TestData")]
    [TestClass()]
    public class GradedTests
    {

        [TestMethod(), Timeout(300)]
        public void SolveTest_Q1MaximumGold()
        {
            RunTest(new Q1MaximumGold("TD1"));
        }

        [TestMethod(), Timeout(300)]
        public void SolveTest_Q2PartitioningSouvenirs()
        {
            RunTest(new Q2PartitioningSouvenirs("TD2"));
        }


        [TestMethod(), Timeout(300)]
        public void SolveTest_Q3MaximizingArithmeticExpression()
        {
            RunTest(new Q3MaximizingArithmeticExpression("TD3"));
        }

        public static void RunTest(Processor p)
        {
            TestTools.RunLocalTest("A7", p.Process, p.TestDataName, p.Verifier);
        }
    }
}
```

۱ Maximum Amount of Gold

فرض کنید مجموعه ای از شمش های طلا را به شما داده ایم و شما باید تا جایی که امکان دارد شمش های طلا را در کیفتان قرار دهید. فقط یک کپی از هر شمش وجود دارد و هر شمش را شما می توانید بردارید یا نه (بنابراین شما نمی توانید کسری از شمش را بردارید).

اگر n تا شمش طلا داشته باشیم با Dynamic Programming بیشترین وزن ممکن از طلا ها که در کیفی به ظرفیت w جا می شود، را پیدا کنید.

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

namespace A7
{
    public class Q1MaximumGold : Processor
    {
        public Q1MaximumGold(string testDataName) : base(testDataName) { }

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

        public long Solve(long W, long[] goldBars)
        {
            //Write your code here
            throw new NotImplementedException();
        }
    }
}
```

۲ Partitioning Souvenirs

شما و دو نفر از دوستانتان پس از بازدید از کشورهای مختلف، به خانه بازگشته اید. حالا شما می خواهید تمام سوغاتی هایی که سه نفری خریداری کرده اید را به طور مساوی بین همدیگر تقسیم کنید. در فایل ورودی این سوال، در خط اول تعداد سوغاتی ها و در خط بعدی مقدار سوغاتی ها خواهد بود. اگر سوغاتی ها را می توان به صورت مساوی تقسیم کرد در فایل خروجی عدد ۱ و در غیر این صورت عدد صفر خواهد بود. به مثال های زیر توجه کنید.

ورودی نمونه	خروجی نمونه
4 3 ,3 ,3 ,3	0

ورودی نمونه	خروجی نمونه
11 17 ,59 ,34 ,57 ,17 ,23 ,67 ,1 ,18 ,2 ,59	1 34 + 67 + 17 = 23 + 59 + 1 + 17 + 18 = 59 + 2 + 57.

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

namespace A7
{
    public class Q2PartitioningSouvenirs : Processor
    {
        public Q2PartitioningSouvenirs(string testDataName)
        : base(testDataName) { }

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

        public long Solve(long souvenirsCount, long[] souvenirs)
        {
            throw new NotImplementedException();
        }
    }
}
```

۳ Maximum Value of an Arithmetic Expression

در این سوال، شما باید با روش Dynamic Programing به گونه ای پرانتز ها را به یک عبارت ریاضی اضافه کنید که مقدار عبارت ریاضی به حداکثر برسد. خط اول فایل ورودی شامل یک رشته s با طول $2n+1$ است که n از یک تا ۱۴ می باشد. رشته را می توانیم به صورت زیر نمایش دهیم:

$$s_0, s_1, s_2, ..., s_n$$

کاراکتر موجود در موقعیت های زوج رشته ی s یک رقم می باشد و کاراکتر موجود در موقعیت های فرد رشته ی s سه عمل +، - و * می باشد.

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

namespace A7
{
    public class Q3MaximizingArithmeticExpression : Processor
    {
        public Q3MaximizingArithmeticExpression(string testDataName)
        : base(testDataName) { }

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

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