



دانشگاه علم و صنعت ایران

دانشکده مهندسی کامپیوتر

ساختمان داده

تمرین ۷*

زهرا حسینی
سهراب نمازی
سید صالح اعتمادی

نیمسال اول ۹۹-۰۰

hosseini_zahra@comp.iust.ac.ir sohrab_namazi@comp.iust.ac.ir	ایمیل/تیمز
fb_A7	نام شاخه
A7	نام پروژه/پوشه/پول ریکوست
۹۹/۹/۱	مهلت تحویل

*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.
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.

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، در غیر اینصورت ممکن است تست های شما در زمان داده شده پاس نشوند.

```

1 using Microsoft.VisualStudio.TestTools.UnitTesting;
2 using TestCommon;
3
4 namespace A7.Tests
5 {
6     [DeploymentItem("TestData")]
7     [TestClass()]
8     public class GradedTests
9     {
10
11         [TestMethod(), Timeout(300)]
12         public void SolveTest_Q1MaximumGold()
13         {
14             RunTest(new Q1MaximumGold("TD1"));
15         }
16
17         [TestMethod(), Timeout(300)]
18         public void SolveTest_Q2PartitioningSouvenirs()
19         {
20             RunTest(new Q2PartitioningSouvenirs("TD2"));
21         }
22
23
24         [TestMethod(), Timeout(300)]
25         public void SolveTest_Q3MaximizingArithmeticExpression()
26         {
27             RunTest(new Q3MaximizingArithmeticExpression("TD3"));
28         }
29
30         public static void RunTest(Processor p)
31         {
32             TestTools.RunLocalTest("A7", p.Process, p.TestDataName, p.Verifier);
33         }
34     }
35 }

```

Maximum Amount of Gold ۱

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.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using TestCommon;
5
6 namespace A7
7 {
8     public class Q1MaximumGold : Processor
9     {
10         public Q1MaximumGold(string testDataName) : base(testDataName) { }
11
12         public override string Process(string inStr) =>
13             TestTools.Process(inStr, (Func<long, long[], long>)Solve);
14
15         public long Solve(long W, long[] goldBars)
16         {
17             //Write your code here
18             throw new NotImplementedException();
19         }
20     }
21 }
```

Partitioning Souvenirs ۲

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 :

4

3,3,3,3

خروجی :

0

ورودی:

11

17, 59, 34, 57, 17, 23, 67, 1, 18, 2, 59

خروجی:

1

$34 + 67 + 17 = 23 + 59 + 1 + 17 + 18 = 59 + 2 + 57.$

```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using TestCommon;
5
6 namespace A7
7 {
8     public class Q2PartitioningSouvenirs : Processor
9     {
10         public Q2PartitioningSouvenirs(string testDataName)
11             : base(testDataName) { }
12
13         public override string Process(string inStr) =>
14             TestTools.Process(inStr, (Func<long, long[], long>)Solve);
15
16         public long Solve(long souvenirsCount, long[] souvenirs)
17         {
18             throw new NotImplementedException();
19         }
20     }
21 }
```

Maximum Value of an Arithmetic Expression ۳

در این سوال، شما باید با روش Dynamic Programming به گونه ای پراگماتیک ترها 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:

$s_0, s_1, s_2, \dots, s_n$

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.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using TestCommon;
5
6 namespace A7
7 {
8     public class Q3MaximizingArithmeticExpression : Processor
9     {
10         public Q3MaximizingArithmeticExpression(string testDataName)
11             : base(testDataName) { }
12
13         public override string Process(string inStr) =>
14             TestTools.Process(inStr, (Func<string, long>)Solve);
15
16         public long Solve(string expression)
17         {
18             throw new NotImplementedException();
19         }
20     }
21 }
```