تمرین ۴ درس ساختمان داده

مریم سادات هاشمی
سید صالح اعتمادی

دانشگاه علم و صنعت ۹۸_۹۷

لطفا به نکات زیر توجه کنید:

- مهلت ارسال این تمرین شنبه ۲۸ 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 "A۴" 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 ۳ 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،۶ سوال دارد که باید به همه ی این سوال ها پاسخ دهید.
ابتدا مانند تمرین های قبل، یک پروژه به نام A۴ 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 شش 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 دستورgit Pull دریافت کنید . در این تمرین تابع process را در TestCommon برای شما پیاده سازی کرده ایم و تنها کافی است برای تابع process هر سوال از تابع process موجود در Deligate TestCommon، کنید. برای مثال برای سوال اول به صورت زیر می شود. همین کار را برای سایر سوال ها انجام دهید.
به شکل زیر دقت کنید.

```
public static string ProcessChangingMoney1(string inStr) =>
    TestTools.Process(inStr, (Func<long,long>) ChangingMoney1);
```

متد RunLocalTest سه ورودی می گیرد:
ورودی اول: نام assignment است. بنابراین برای این تمرین، این پارامتر A۴ خواهد بود.
ورودی دوم: 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.
ورودی سوم : 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 Money Change

در این سوال، قرار است شما یک الگوریتم حریصانه که توسط صندوقداران در سراسر جهان میلیون ها بار در روز استفاده می شود، را طراحی و پیاده سازی کنید.
فرض کنید سه سکه با مقدار های ۱، ۵ و ۱۰ در اختیار دارید. حداقل تعداد سکه هایی که لازم است تا مقدار پول ورودی را با سکه های مذکور بسازید؛ چقدر است؟ برای حل این سوال الگوریتم حریصانه ای را پیاده سازی کنید.

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

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

- Algorithm Function : ChangingMoney1
- Process Function : ProcessChangingMoney1
- Test Function : ChangingMoney1Test

```
public static long ChangingMoney1(long money)
{
    // your code here
    return 0;
}
```

```
public static string ProcessChangingMoney1(string inStr) =>
    TestTools.Process(inStr, (Func<long,long>) ChangingMoney1);
```

```
[TestMethod(), Timeout(1000)]
[DeploymentItem(@"TestData", "A4_TestData")]
0 references | Sauleh Eetemadi, 1 day ago | 1 author, 3 changes
public void ChangingMoney1Test()
{
    TestTools.RunLocalTest("A4", Program.ProcessChangingMoney1, "TD1");
}
```

۲ Maximum Value of the Loot

یک دزد غنیمت هایی پیدا کرده است که از طرفیت کیسه اش بیشتر است. به او برای پیدا کردن با ارزش ترین ترکیب از غنیمت ها کمک کنید. فرض کنید که هر کسری از یک غنیمت را می تواند در کیسه خود قرار دهد. برای مثال اگر ظرفیت کیسه ی دزد ۱۰ باشد و یک عنمیت با ارزش ۳۰۰ و ظرفیت ۳۰ وجود داشته باشد؛ دزد می تواند کسری از غنیمت که ظرفیتش ۱۰ و ارزشش ۱۰۰ می باشد را بردارد و در کیسه ی خود بگذارد. الگوریتم خود را به صورت حریصانه بنویسید.(خط اول testcase ها ظرفیت کیسه ی دزد و خط های بعدی به ترتیب عدد اول وزن آیتم و عدد دوم ارزش آن است)

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

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

- Algorithm Function : MaximizingLoot2
- Process Function : ProcessMaximizingLoot2
- Test Function : MaximizingLoot2Test

```
public static long MaximizingLoot2(
    long capacity, long[] weights, long[] values)
{
    // your code here
    return 0;
}
```

```
public static string ProcessMaximizingLoot2(string inStr) =>
    TestTools.Process(inStr,
        (Func<long, long[], long[], long>) MaximizingLoot2);
```

```
[TestMethod(), Timeout(1000)]
[DeploymentItem(@"TestData", "A4_TestData")]
0 references | Sauleh Eetemadi, 1 day ago | 1 author, 3 changes
public void MaximizingLoot2Test()
{
    TestTools.RunLocalTest("A4", Program.ProcessMaximizingLoot2, "TD2");
}
```

۳ Maximum Advertisement Revenue

شما n عدد آگهی برای قرار دادن در یک صفحه اینترنتی محبوب را دارید. برای هر آگهی، شما می دانید چه مقدار هزینه قرار است آگهی دهنده به ازای یک کلیک در این آگهی پرداخت کند. شما در صفحه خود n تا slot را تنظیم کرده اید و تعداد کلیک های مورد انتظار را برای هر slot در هر روز تخمین زده اید. شما باید الگوریتمی طراحی کنید که آگهی ها را میان slot ها به گونه ای توزیع کندکه درآمد کل به حداکثر برسد. دقت کنید که در خط اول هر testcase تعداد slot ها یا آگهی ها می باشد. سپس در خط های بعدی عدد اول هزینه ی یک کلیک بر روی آگهی i ام است و عدد دوم میانگین تعداد کلیک هایی در روز است که بر روی $slot_i$ ام انجام می شود. بنابراین شما هزینه ی کلیک بر روری یک آگهی را در یک آرایه مانند a_i و میانگین تعداد کلیک ها در روز را در آرایه دیگری مانند b_i بریزید. سپس این دو آرایه را به همراه تعداد slot ها یعنی n به عنوان ورودی به تابع الگوریتم خود بدهید.

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

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

- Algorithm Function : MaximizingOnlineAdRevenue3
- Process Function : ProcessMaximizingOnlineAdRevenue3
- Test Function : MaximizingOnlineAdRevenue3Test

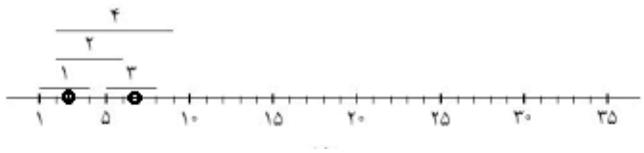
```
public static long MaximizingOnlineAdRevenue3(long slotCount,
    long[] adRevenue, long[] averageDailyClick)
{
    // your code here
    return 0;
}
```

```
public static string ProcessMaximizingOnlineAdRevenue3(string inStr) =>
    TestTools.Process(inStr,
        (Func<long, long[], long[], long>) MaximizingOnlineAdRevenue3);
```

```
[TestMethod(), Timeout(1000)]
[DeploymentItem(@"TestData", "A4_TestData")]
0 references | Sauleh Eetemadi, 1 day ago | 1 author, 1 change
public void MaximizingOnlineAdRevenue3Test()
{
    TestTools.RunLocalTest("A4", Program.ProcessMaximizingOnlineAdRevenue3, "TD3");
}
```

۴ Collecting Signatures

شما مسئول جمع آوری امضا از همه مستاجران یک ساختمان خاص هستید. برای هر مستاجر، شما می دانیدکه در چه بازه ی زمانی در خانه است. شما باید تمام امضا ها را تا جایی که ممکن است با کمترین تعداد مراجعه به ساختمان، جمع آوری کنید. مدل ریاضی برای این مشکل به صورت شکل زیر است. شما مجموعه ای از بازه ها را در یک خط قرار داده اید و هدف شما این است که کمترین تعداد نقطه ها را بر روی خط پیدا کنید که هر بازه حداقل شامل یک نقطه باشد. دقت کنید که در خط اول هر testcase تعداد بازه ها می باشد و در خط های بعدی، به ترتیب عدد اول شروع بازه و عدد دوم پایان بازه می باشد. بنابراین شما شروع بازه ها را در یک آرایه مانند a_i و پایان بازه ها را در آرایه دیگری مانند b_i بریزید. سپس این دو آرایه را به همراه تعداد بازه ها یعنی n به عنوان ورودی به تابع الگوریتم خود بدهید.



- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

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

- Algorithm Function : CollectingSignatures4
- Process Function : ProcessCollectingSignatures4
- Test Function : ProcessCollectingSignatures4Test

```
public static long CollectingSignatures4(long tenantCount,
    long[] startTimes, long[] endTimes)
{
    // your code here
    return 0;
}
```

```
public static string ProcessCollectingSignatures4(string inStr) =>
    TestTools.Process(inStr,
        (Func<long, long[], long[], long>)CollectingSignatures4);
```

```
[TestMethod(), Timeout(1000)]
[DeploymentItem(@"TestData", "A4_TestData")]
0 references | Sauleh Eetemadi, 1 day ago | 1 author, 1 change
public void ProcessCollectingSignatures4Test()
{
    TestTools.RunLocalTest("A4", Program.ProcessCollectingSignatures4, "TD4");
}
```

۵ Maximum Number of Prizes

فرض کنید قرار است شما یک رقابت هیجان انگیز را برای کودکان سازماندهی کنید.شما یک صندوق جایزه دارید که n آب نبات در آن وجود دارد. شما باید این آبنبات ها را به بچه هایی که در مکان های یک تا k در مسابقه قرار گرفته اند؛ به عنوان جایزه بدهید؛ به گونه ای که رتبه های بالاتر تعداد بیشتری آبنبات بگیرند. شما باید k را به گونه ای پیدا کنید که تا جای ممکن تعداد بیشتری از بچه ها جایزه بگیرند و خوشحال شوند.در واقع مدل ریاضی این سول به این صورت است که یک عدد صحیح مثبت n را به صورت جمعی از یک سری اعداد صحیح مثبت که دو به دو نیز متمایزهستند؛ دربیاوریم به گونه ای که تعداد این اعداد بیشترین مقدار ممکن باشد.

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

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

- Algorithm Function : MaximizeNumberOfPrizePlaces5
- Process Function : ProcessMaximizeNumberOfPrizePlaces5
- Test Function : MaximizeNumberOfPrizePlaces5Test

```
public static long[] MaximizeNumberOfPrizePlaces5(long n)
{
    // your code here
    return new long[] { 0 };
}
```

```
public static string ProcessMaximizeNumberOfPrizePlaces5(string inStr) =>
    TestTools.Process(inStr, (Func<long, long[]>) MaximizeNumberOfPrizePlaces5);
```

```
[TestMethod(), Timeout(1000)]
[DeploymentItem(@"TestData", "A4_TestData")]
0 references | Sauleh Eetemadi, 1 day ago | 1 author, 1 change
public void MaximizeNumberOfPrizePlaces5Test()
{
    TestTools.RunLocalTest("A4", Program.ProcessMaximizeNumberOfPrizePlaces5, "TD5");
}
```

۶ Maximum Salary

فرض کنید برای استخدام در یک مصاحبه شرکت کرده اید و به عنوان آخرین سوال مصاحبه، رئیس شما به شما چند قطعه کاغذ با اعداد روی آن می دهد و از شما می خواهد بزرگترین شماره را از این اعداد بنویسید. پاسخ شما همان حقوق شما خواهد بود، بنابراین شما بسیار علاقه مند به حداکثر رساندن این عدد هستید. چطور می توانید این کار را بکنید؟ در ویدئو های درس، الگوریتم زیر را برای ساختن بزرگترین عدد از شماره های تک رقمی داده شده در نظر گرفتیم.

متاسفانه این الگوریتم تنها در صورتی کار می کند که ورودی از اعداد تک رقمی باشد. به عنوان مثال، برای یک ورودی متشکل از دو عدد صحیح ۲۳ و ۳ (۲۳ عدد یک رقمی نیست!) الگوریتم عدد ۳۳۳ را برمی گرداند، در حالی که بزرگترین عدد در واقع ۳۲۳ است. به عبارت دیگر، استفاده از بزرگترین عدد از ورودی به عنوان شماره اول همیشه ما را به جواب درست نمی رساند. هدف شما در این مشکل این است که الگوریتم بالا را بهینه سازی کنید تا کارایی آن نه تنها برای عدد تک رقمی، بلکه برای هر عدد صحیح دلخواه مثبت باشد.

```
LargestNumber(Digits):
answer ← empty string
while Digits is not empty:
  maxDigit ← −∞
  for digit in Digits:
    if digit ≥ maxDigit:
      maxDigit ← digit
  append maxDigit to answer
  remove maxDigit from Digits
return answer
```

- محدودیت زمانی : **۱۰۰۰ میلی ثانیه**

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

- Algorithm Function : MaximizeSalary6
- Process Function : ProcessMaximizeSalary6
- Test Function : MaximizeSalary6Test

```
public static string MaximizeSalary6(long n, long[] numbers)
{
    //write your code here
    return "0";
}
```

```
public static string ProcessMaximizeSalary6(string inStr) =>
    TestTools.Process(inStr, MaximizeSalary6);


[TestMethod(), Timeout(1000)]
[DeploymentItem(@"TestData", "A4_TestData")]
0 references | Sauleh Eetemadi, 1 day ago | 1 author, 1 change
public void MaximizeSalary6Test()
{
    TestTools.RunLocalTest("A4", Program.ProcessMaximizeSalary6, "TD6");
}
```