برنامه‌سازی پیشرفته

تمرین کامپیوتری شماره ۳

مدرس: رامتین خسروی

طراحان: علی پادیاو، علی حمزه‌پور، الهه خداوردی، امیرعلی رحیمی، مبینا مهرآذر، مهدی نوری، پریسا یحیی‌پور

مهلت تحویل: سه‌شنبه ۲۱ فروردین ۱۴۰۳، ساعت ۲۳:۵۵

مقدمه

هدف از این تمرین آشنایی شما با طراحی بالا به پایین[1] یک مسئله است. با توجه به حجم پروژه لازم است که قبل از شروع پیاده‌سازی زمانی را به طراحی اختصاص دهید. در غیر این صورت در هنگام پیاده‌سازی با مشکل مواجه می‌شوید. بنابراین ابتدا به چگونگی شکستن این مسئله به مسائل کوچک‌تر و پخش کردن مسئولیت‌ها میان قسمت‌های مختلف برنامه فکر کنید.

برای آشنایی بیشتر شما با این نوع طراحی می‌توانید به ویدیویی که در بخش محتوای دستیاران آموزشی در صفحه درس قرار گرفته مراجعه کنید.

یوتی‌پِی

احتمالا با برنامه Splitwise آشنا هستید. با استفاده از این برنامه می‌توانید در برنامه‌های دسته‌جمعی نظیر سفر با دوستانتان، هزینه‌های سفر را به راحتی مدیریت کنید. در این برنامه، هر فرد خرج‌هایی که انجام داده را وارد کرده و سهم هر فرد از آن خرج را مشخص می‌کند. در نهایت با توجه به الگوریتم‌های برنامه، بدهی‌ها ساده‌سازی شده و بدهی هر فرد به افراد دیگر مشخص می‌شود. در این تمرین قصد پیاده‌سازی یک نمونه ساده‌شده از این برنامه را داریم.

[1] Top-Down Design

قالب فایل‌های ورودی

اطلاعات افراد و اطلاعات مربوط به هزینه‌ها در دو فایل جداگانه قرار دارند. مسیر این فایل‌ها به ترتیب (ابتدا مسیر فایل کاربران سپس مسیر فایل خرج‌ها) از طریق آرگومان خط فرمان به برنامه داده می‌شود. برای آشنایی با این نوع آرگومان‌ها، می‌توانید این لینک را مشاهده کنید. نوع فایل‌های ورودی به صورت CSV[2] هستند و برای آشنایی با این نوع فایل‌ها می‌توانید این لینک را بررسی کنید.

نمونه ورودی

```
./UTPay.out </path/to/users/file> </path/to/expenses/file>
```

فایل کاربران

در این فایل یک ستون name وجود دارد. اسامی افراد در ردیف‌های این ستون قرار دارد.

نمونه فایل کاربران

```
name
Ali
Parisa
Amir Ali
Elahe
Mobina
Mahdi
```

فایل خرج‌ها

در هر سطر یک هزینه تعریف شده است. این فایل شامل سه ستون به ترتیب type,payers,borrowers می‌باشد. در صورتی که چندین قرض‌گیرنده یا پرداخت‌کننده وجود داشته باشد نام آن‌ها با سمی‌کالن[3] (;) جدا می‌شود. داده مربوط به هر فرد (در صورت وجود) با دو نقطه (:) مشخص می‌شود.

[2] Comma-Separated Values

[3] Semicolon

نمونه فایل خرج‌ها

```
type,payers,borrowers
equally,Ali:250,
equally,Mobina:380,Elahe;Amir Ali;Mahdi;Ali
unequally,Elahe:530,Mahdi:130;Parisa:210;Ali:190
adjustment,Parisa:560;Mahdi:100,Mobina:20;Ali:40
```

تذکر: در تقسیم هزینه بین افراد، سهم هر فرد به جز نفر آخر را تا ۲ رقم اعشار محاسبه و رو به پایین گرد کنید. سهم نفر آخر به گونه‌ای محاسبه شود که جمع تمام سهم‌ها با هزینه کل برابر شود. آخرین نفر به دو صورت مشخص می‌شود:

1. **اگر اسمی در قسمت borrowers وجود نداشته باشد -> آخرین نفر فایل کاربران**
2. **در قسمت borrowers نام حداقل یک نفر ذکر شده باشد -> آخرین نفر قسمت borrowers**

به طور کلی ۳ نوع[4] تقسیم‌بندی داریم که هر کدام در ادامه توضیح داده می‌شوند:

۱. به صورت مساوی (Equally)

در این تقسیم‌بندی هزینه پرداخت‌شده توسط پرداخت‌کننده به طور مساوی بین افرادی که در هزینه شریک بودند تقسیم می‌شود.

در صورتی که بعد از هزینه کلی نام قرض گیرنده‌ها نیاید، هزینه به طور مساوی بین همه افراد (شامل فرد پرداخت‌کننده) تقسیم می‌شود.

توجه کنید شخص پرداخت‌کننده در دو حالت در هزینه کل سهم دارد:

1. نام او در بین افراد قرض‌گیرنده بیاید.
2. هزینه باید بین تمام افراد گروه تقسیم شود.

قالب در فایل خرج‌ها

```
equally,<payer1>:<cost1>;<payer2>:<cost2>;...,<borrower1>;<borrower2>
;...
```

[4] Type

نمونه در فایل خرج‌ها

```
equally,Ali:250,
equally,Mobina:380,Elahe;Amir Ali;Mahdi;Ali
```

توضیح مثال اول: ۲۵۰ تومان بین تمام افراد تقسیم می‌شود. سهم همه ۴۱.۶۶ می‌شود و سهم مهدی که آخرین نفر است، ۴۱.۷ می‌باشد.
توضیح مثال دوم: ۳۸۰ تومان باید بین الهه، امیرعلی، مهدی و علی تقسیم شود. سهم هر فرد برابر ۹۵ تومان خواهد بود. توجه کنید چون مبینا در بین قرض‌گیرنده‌ها نیست، سهمی به او تعلق نمی‌گیرد و فقط پرداخت‌کننده است.

۲. به صورت غیر مساوی (Unequally)

در این تقسیم‌بندی سهم هر فرد از کل هزینه پرداخت شده، مشخص می‌شود.

قالب در فایل خرج‌ها

```
unequally,<payer1>:<cost1>;<payer2>:<cost2>;...,<borrower1>:<amount1>
;<borrower2>:<amount2>;...
```

نمونه در فایل خرج‌ها

```
unequally,Elahe:530,Mahdi:130;Parisa:210;Ali:190
```

توضیح مثال: هزینه کلی ۵۳۰ تومان است و پریسا و علی و مهدی در این هزینه سهم دارند. در کنار نام فرد سهم او آمده است.
تضمین می‌شود مجموع سهم افراد، برابر با هزینه کل خواهد بود.

۳. به صورت تنظیم (Adjustment)

در این تقسیم‌بندی افرادی که سهم اضافی دارند به همراه مقدار سهم اضافی‌شان مشخص و باقیمانده هزینه بین **همه افراد** به صورت مساوی تقسیم می‌شود.

قالب در فایل خرج‌ها

```
adjustment,<payer1>:<cost1>;<payer2>:<cost2>;...,<borrower1>:<extra_amount1>;<borrower2>:<extra_amount2>;...
```

نمونه در فایل خرج‌ها

```
adjustment,Parisa:560;Mahdi:100,Mobina:20;Ali:40
```

توضیح مثال: کل هزینه برابر با ۶۶۰ تومان (۵۶۰ تومان برای پریسا و ۱۰۰ تومان برای مهدی) است و در مجموع ۶۰ تومان هزینه اضافی داریم. بنابراین ۶۰۰ تومان باید بین تمام افراد گروه تقسیم شود. سهم هر شخص ۱۰۰ تومان می‌شود. اما چون علی و مبینا سهم اضافی دارند، خواهیم داشت:

- پریسا، مهدی، امیرعلی، الهه: هر کدام ۱۰۰ تومان
- علی: ۱۴۰ تومان
- مبینا: ۱۲۰ تومان

بهینه‌سازی

پس از محاسبه کل مقدار پولی که هر شخص باید پرداخت یا دریافت کند (مقدار دریافتی را مثبت و مقدار پرداختی را منفی در نظر بگیرید)، قصد داریم به گونه‌ای عمل کنیم که تعداد واریزها کاهش یابد. برای مثال مقدار پرداخت و دریافت به شکل زیر خواهد بود:

مثال دریافت‌ها و پرداخت‌ها

```
Mobina: 218.34
Ali: -216.66
Mahdi: -266.7
Parisa: 208.34
Amir Ali: -236.66
Elahe: 293.34
```

برای این کار بدهکاران و طلبکاران را به صورت نزولی و بر اساس بدهی/طلب مرتب می‌کنیم (در صورتی که مقدار بدهی/طلب دو نفر یکسان بود بر اساس اسم مرتب کنید) و از شخصی که بیشترین بدهی را دارد شروع می‌کنیم. اولین بدهکار باید به فردی که بیشترین طلب را دارد پرداخت کند. در صورتی که بدهی فرد از طلب دیگری بیشتر باشد، مقدار اضافی، به طلبکار بعدی پرداخت می‌شود. در صورتی هم که بدهی فرد از طلب دیگری کمتر باشد، سراغ بدهکار بعدی می‌رویم. برای مثال، الهه بیشترین طلب و مهدی بیشترین بدهی را دارد. مهدی ۲۶۶.۷ تومان به الهه می‌دهد. باقیمانده طلب الهه که برابر ۲۶.۶۴ تومان است توسط شخص بعدی یعنی امیرعلی پرداخت می‌شود. ۲۱۰.۰۲ تومان از بدهی امیرعلی می‌ماند که به مبینا پرداخت می‌شود.

در صورتی که هنگام انتخاب بدهکار یا طلبکار، مقدار بدهی/طلب چند نفر یکسان باشد، این افراد را بر اساس نامشان و به صورت صعودی مرتب کنید. بدهی/طلب فردی که در لیست مرتب‌شده زودتر آمده باشد، اولویت بیشتری دارد.

قالب خروجی

پس از انجام محاسبات مورد نیاز، مقدار پولی که هر شخص باید به دیگری بدهد را به صورت زیر چاپ کنید.

ترتیب چاپ بر اساس مقدار پرداختی به صورت نزولی باشد. در صورتی که مقدار پرداختی یکسان باشد، بر اساس نام بدهکار[5] و در صورتی که نام بدهکار هم یکسان باشد بر اساس نام طلبکار[6] و به صورت صعودی مرتب کنید.

قالب خروجی

```
<debtor1> -> <creditor1>: <amount1>
<debtor2> -> <creditor2>: <amount2>
...
```

نمونه خروجی

```
Mahdi -> Elahe: 266.7
Amir Ali -> Mobina: 210.02
Ali -> Parisa: 208.34
Amir Ali -> Elahe: 26.64
Ali -> Mobina: 8.32
```

[5] Debtor

[6] Creditor

نحوه مرتب‌سازی بر اساس نام

در زبان C++، هر رشته شامل تعدادی کاراکتر است و هر کاراکتر دارای یک کد ASCII است. این کد برای حروف انگلیسی کوچک از ۹۷ تا ۱۲۲ و برای حروف انگلیسی بزرگ از ۶۵ تا ۹۰ است. جدول کامل ASCII را می‌توانید در این لینک مشاهده کنید.

با استفاده از این روش کدگذاری، می‌توانیم یک روش مقایسه رشته‌ها را به صورت مقایسه این کدها با یکدیگر در نظر بگیریم. این روش به این صورت است که کد ASCII کاراکترهای اول دو رشته با همدیگر مقایسه می‌شوند، در صورتی که یکی از آن‌ها کوچک‌تر از دیگری باشد، کل آن رشته کوچک‌تر از رشته دیگر در نظر گرفته می‌شود. در صورتی که کاراکترهای اول دو رشته با همدیگر برابر باشند، کاراکترهای دوم را مقایسه می‌کنیم. این کار را تا زمانی انجام می‌دهیم که به کاراکتری برسیم که در دو رشته یکسان نباشد. در صورتی که پیش از رسیدن به چنین کاراکتری به انتهای هر دو رشته برسیم، دو رشته با همدیگر برابر در نظر گرفته می‌شوند. در زبان C++، اپراتورهای مقایسه‌ای نیز برای رشته‌ها دقیقا به همین صورت پیاده‌سازی شده‌اند.

در این تمرین، در بخش‌هایی که خواسته شده آیتم‌ها را بر اساس نام مرتب کنید، لازم است دقیقا به همین صورت عمل کنید. برای مثال نام‌های زیر را در نظر بگیرید:

نمونه نام‌ها
`Amir, amin, Borna, dorsa, AMirreza`

در صورت مرتب‌سازی، نام‌ها به صورت زیر خواهند بود:

نمونه نام‌های مرتب‌شده
`AMirreza, Amir, Borna, amin, dorsa`

نکات و نحوهٔ تحویل

- کد خود را در قالب یک فایل با نام A3-SID.cpp در صفحهٔ eLearn درس بارگذاری کنید که SID شمارهٔ دانشجویی شماست؛ برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۰۲۰۰۰ باشد، نام پروندهٔ شما باید A3-810102000.cpp باشد که شامل کد شما است. **اشتباه در نام فایلِ تحویل‌داده‌شده می‌تواند منجر به کسر نمره از شما شود.**
- برنامهٔ شما باید در سیستم عامل لینوکس و با مترجم g++ با استاندارد c++20 ترجمه و در زمان معقول برای ورودی های آزمون اجرا شود.
- در این تمرین اجازه استفاده از شیءگرایی و makefile را ندارید.
- درستی برنامهٔ شما از طریق آزمون‌های خودکار سنجیده می‌شود؛ بنابراین پیشنهاد می‌شود که با استفاده از ابزارهایی مانند diff خروجی برنامه خود را با خروجی‌هایی که در اختیارتان قرار داده شده است مطابقت دهید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

نمرات

- تمیزی کد
  - رعایت کردن نام‌گذاری صحیح و انسجام[7]
  - عدم وجود کد تکراری
  - رعایت دندانه‌گذاری[8]
  - عدم استفاده از متغیرهای گلوبال
  - استفاده **صحیح** از متغیرهای ثابت به جای Magic Value-ها
- درستی کد
  - آزمون‌های خودکار
- طراحی
  - شکستن مناسب و مرحله به مرحله مسئله
  - ذخیره اطلاعات در ساختار داده‌های مناسب
  - ساختاردهی کد در قالب توابع کوتاه که فقط یک کار را انجام می‌دهند

[7] Consistency

[8] Indentation

دقت کنید که موارد ذکر شده لزوما کل نمره شما را تشکیل نمی‌دهند و ممکن است با تغییراتی همراه باشند.