برنامه‌سازی پیشرفته

تمرین شماره‌ی یک

مدرس: رامتین خسروی

طراحان: سپهر سامنی، محمد مهدوی‌دوست

موعد تحویل: سه‌شنبه ۲۷ بهمن ۱۳۹۴

غُل‌کوین

در این تمرین قصد داریم نسخه‌ی بسیار ساده‌ای از بیت‌کوین را طراحی و پیاده‌سازی کنیم.

بیت‌کوین، نوعی پول دیجیتال بدون پشتوانه است که در سال ۲۰۰۸ میلادی توسط ساتوشی ناکاموتو معرفی و در سال ۲۰۰۹ به صورت متن‌باز منتشر شد. ریشه‌های پول دیجیتال به حبابِ دات‌کام[1] در دهه‌ی ۱۹۹۰ میلادی باز می‌گردد. در این دوره، سرویس‌های پرداخت مبتنی بر پول دیجیتال به کاربران اجازه می‌دادند تا پول فیزیکی خود را به پول دیجیتال تبدیل کرده و آن را آزادانه و بدون هیچ نظارتی بین یک‌دیگر رد و بدل کنند. از نخستین ارائه‌دهندگان این سرویس‌ها، می‌توان به ای‌گلد[2] و رزرو آزادی[3] اشاره کرده که خدمات خود را به صورت متمرکز ارائه می‌دادند. هر دوی این سرویس‌ها به استفاده برای پول‌شویی شهرت داشته و در نهایت تعطیل شدند. بیت‌کوین، اولین پول دیجیتال غیرمتمرکز است؛ به این معنی که به کاربران این امکان را می‌دهد، بدون هیچ واسطه‌ای انتقال پول را به صورت غیرقابل بازگشت انجام دهند. این سامانه، کنترل‌کننده‌ی متمرکز ندارد، و توسط هیچ نهاد دولتی، سازمان، و یا مؤسسه‌ای اداره نمی‌شود. برای کسب اطلاعات بیش‌تر در مورد بیت‌کوین می‌توانید به صفحه ویکی آن مراجعه کنید.

اطلاعات کاربرانِ سامانه شامل نام، شناسه‌ی یکتا، و سرمایه‌ی اولیه در ابتدای برنامه در اختیار شما قرار خواهد گرفت. برای ایجاد هر تراکنش نیز، شناسه‌ی فرستنده، شناسه‌ی گیرنده، مبلغ، و زمان انجام تراکنش را مشخص خواهیم کرد. سرمایه‌ی اولیه کاربران و مبلغ هر تراکنش، هر دو ضریبی از یک بیت‌کوین است. مجموعه‌ای از تراکنش‌ها را نیز در یک بلاک نگهداری می‌کنیم. به بیان دقیق‌تر، هر بلاک شامل تعداد نامشخصی تراکنش خواهد بود که به صورت مرتب (بر اساس زمان تراکنش) نگهداری می‌شوند.

این سرویس از زنجیره‌ای از بلاک‌ها تشکیل شده‌است. با بسته شدن یک بلاک تراکنش، یک بلاک جدید ایجاد خواهد شد و تراکنش‌ها نیز از این لحظه به بعد به بلاک جدید اضافه خواهند شد. در نظر داشته باشید که اضافه شدن موفقیت‌آمیز یک تراکنش تنها زمانی امکان‌پذیر است که آن تراکنش از نظر زمانی صحیح باشد، یعنی زمان اجرای آن بعد از زمان اجرای آخرین تراکنش بلاک قبلی باشد. با بسته شدن هر بلاک، یک فرآیند دیگر نیز صورت می‌گیرد: تابعی پیچیده برای اطمینان از درستی محاسبه‌ی تراکنش‌ها روی کل زنجیره اجرا خواهد شد. چگونگی کار این تابع، در ادامه بررسی خواهد شد. توجه کنید که بلاک جدیدی که برای تراکنش‌های آتی استفاده خواهد شد، پس از خاتمه این تابع ایجاد می‌شود.

با توجه به اینکه این تمرین، نخستین تمرین کامپیوتری شماست، فرآیند طراحی و پیاده‌سازی را به صورت مرحله به مرحله توضیح خواهیم داد و قویاً توصیه می‌کنیم که شما نیز مطابق با همین مراحل، برنامه خود را توسعه دهید.

[1] Dot-com bubble
[2] E-Gold
[3] Liberty Reserve

۱. نگهداری اطلاعات

همان‌طور که در بخش قبل اشاره کردیم، سامانه پرداخت، از تعدادی کاربر تشکیل شده‌است که با یکدیگر تراکنش دارند، این تراکنش‌ها در بلاک‌ها نگهداری می‌شوند و سامانه نیز از زنجیره‌ای از بلاک‌ها تشکیل شده‌است. با استفاده از استراکت‌ها، بردارها[4] و انواع متغیرها، ساختارهای داده مورد نیاز برای نگهداری این اطلاعات را طراحی کنید. در نظر داشته باشید که این طراحی اولیه‌ی شماست و ممکن است کامل نباشد .پس نگران نباشید و در صورتی که نیاز باشد، می‌توانید آن را در ادامه اصلاح کنید. همچنین استفاده از typedef برای افزایش خوانایی برنامه توصیه می‌شود.

۲. ایجاد داده نمونه و تولید خروجی مناسب

در این مرحله قصد داریم ساختارهای داده مرحله قبل را با مقادیر نمونه پر کنیم و چگونگی نمایش صحیح هر یک در خروجی استاندارد را مشاهده کنیم. یک تابع main بنویسید و سه کاربر با نام‌های قمر، سپهر، و مستانه ایجاد کنید که به ترتیب سرمایه‌ی اولیه‌ای برابر با ۱۱۵، ۳۴، و ۲۷۳ بیت‌کوین دارند. تابعی پیاده‌سازی کنید که پس از فراخوانی، لیستی از حساب‌های کاربری را مشابه آن‌چه در شکل زیر نمایش داده شده‌است، در خروجی استاندارد چاپ کند:

```
1           13                                  44      50
UID         Name                                   Balance
--------------------------------------------------
0           Ghamar                                     115
1           Sepehr                                      34
2           Mastaneh                                   273
```

یک تراکنش از قمر به سپهر با مبلغ ۳۰ بیت‌کوین و در زمان ۱۴۵۴۹۴۶۲۷۰ (۸ فوریه ۲۰۱۶ ساعت ۱۹:۱۵) ایجاد کنید. تابعی بنویسید که لیست تراکنش‌های بلاک فعلی را بر اساس زمان انجام (صعودی) مشابه آنچه در شکل زیر نمایش داده شده‌است، در خروجی استاندارد چاپ کند:

```
1                10                  20                                    41                51                            80
Block        :0          Transaction Count     :1          Last Timestamp      :1454946270
--------------------------------------------------------------------------------
1454946270      Ghamar                    Sepehr                                   30
1                17                        43
```

هدف از پیاده‌سازی این دو تابع، استفاده از آنها حین فرآیند توسعه است به این صورت که بلافاصله پس از پیاده‌سازی بخشی از منطق برنامه، وضعیت داده‌ها را مجددا بررسی کنیم و از درستی آخرین تغییرات اعمال شده، اطمینان حاصل کنیم. این تکنیک در جلوگیری از خطاهای احتمالی نیز بسیار موثر است.

۳. تابع main و روند کلی برنامه

در پیاده‌سازی این سرویس باید به این نکته توجه کنیم که ارتباط بین اپراتور و سامانه، ارتباطی تعاملی است، یعنی تا زمانی که اپراتور دستورات مورد نظر خود را در واسط خط فرمان وارد می‌کند (تا زمانی که در ورودی استاندارد داده داریم)، این دستورات مورد رسیدگی قرار می‌گیرند و پاسخ مناسب به هر یک نمایش داده خواهد شد. تصویر زیر روند

---

[4] Vector

کلی برنامه را نمایش می‌دهد و برنامه‌ی نهایی شما، نسخه تکمیل‌شده این شبه‌کد خواهد بود. یادآوری می‌کنیم که پیاده‌سازی کل برنامه در تابع main مورد پذیرش واقع نخواهد شد.

```
int main(int argc, char* argv[]) {

  // local variables
  ...

  // initialize accounts from standard input (details will be explained)
  ...

  while(true) {

    // read command from standard input
    ...

    // parse a given command and process it (you may need to break from the loop)
    ...

  }

  return 0;
}
```

۴. رسیدگی به یک تراکنش

زمانی که اپراتور دستور ایجاد یک تراکنش جدید را وارد می‌کند، رسیدگی به درخواست طی چند مرحله صورت می‌گیرد: در ابتدا برنامه‌ی شما تشخیص می‌دهد که دستور فوق وارد شده‌است. سپس آرگومان‌ها را استخراج کرده و درخواست را اعتبارسنجی می‌کند. درخواست اپراتور تنها زمانی معتبر است که فرستنده و گیرنده در سامانه وجود داشته باشند، فرستنده به اندازه‌ی کافی موجودی داشته باشد و زمان انجام تراکنش نیز معتبر باشد، یعنی از زمان آخرین تراکنش در بلاک قبلی بزرگتر باشد. پس از اطمینان یافتن از این شرایط، برنامه شما یک تراکنش در سامانه ایجاد می‌کند، آن را به بلاک فعلی اضافه می‌کند، موجودی فرستنده و گیرنده را به روزرسانی می‌کند. شما می‌توانید با استفاده از توابع چاپ اطلاعات که در بخش‌های قبل پیاده‌سازی کردید، از اضافه شده تراکنش به بلاک و تغییر موجودی فرستنده و گیرنده اطمینان حاصل کنید.

۵. بسته شدن یک بلاک

زمانی که اپراتور دستور خاتمه یک بلاک را وارد می‌کند، دو فرآیند اصلی در سامانه صورت خواهد گرفت: ابتدا تابعی پیچیده برای اطمینان از درستی محاسبه‌ی تراکنش‌ها روی کل زنجیره اجرا خواهد شد و خروجی آن به اپراتور نمایش داده خواهد شد. در پایان نیز یک بلاک جدید برای تراکنش‌های آتی ایجاد خواهد شد. پیاده‌سازی تابع درستی‌سنجی را به آینده موکول می‌کنیم لذا باقیمانده تغییرات مورد نیاز را اعمال کنید و با استفاده از دستورهای چاپ، از درستی عملکرد برنامه خود اطمینان حاصل کنید.

۶. ایجاد حساب‌های کاربری در ابتدای برنامه

در خط اول ورودی استاندارد، تعداد کاربران سامانه (n) در اختیار شما قرار خواهد گرفت و هر یک از n خط بعدی ورودی، مشخصات یکی از کاربران را با فرمت "Unique Identifier, Name, Initial Capital" مشخص خواهد کرد.

```
3
0, "Ghamar", 115
1, "Sepehr", 34
2, "Mastaneh", 273
...
```

این فرمت CSV نام دارد و یکی از شناخته‌شده‌ترین فرمت‌های نمایش اطلاعات است. تصویر زیر بخشی ابتدایی ورودی نمونه مربوط به کاربران را نمایش می‌دهد. برنامه خود را به گونه‌ای تکمیل کنید که در ابتدای اجرا، مشخصات کاربران را از ورودی استاندارد بخواند و اطلاعات آن‌ها را ذخیره کنید. در نظر داشته باشید که شناسه‌ی یکتا، مقداری عددی است و نام کاربر می‌تواند شامل فاصله (space) نیز باشد. همچنین قبل یا بعد از هر کاراکتر کاما (,) نیز ممکن است بین صفر تا چند فاصله وجود داشته باشد.

۷. پیاده‌سازی تابع درستی‌سنجی

در این مرحله تابع درستی‌سنجی را برای بلاک n ام (n از صفر شروع می‌شود) که در حال بسته‌شدن است پیاده‌سازی خواهیم کرد به این صورت که برای هر یک از تراکنش‌های این بلاک، یک اوکتت[5] معادل بدست می‌آوریم، پارامتر مجهول R0 را بین ۰ تا ۲۵۵ مقدار می‌دهیم و مطلوب‌ترین حالت را به باز می‌گردانیم.

برای محاسبه اوکتت معادل تراکنش k ام (k از صفر شروع می‌شود)، سلسله محاسبات زیر را انجام می‌دهیم:

الف) شناسه‌ی فرستنده‌ی تراکنش را در شناسه‌ی گیرنده ضرب کرده و مقدار حاصل را m می‌نامیم. باقیمانده‌ی m را بر تعداد بلاک‌های زنجیره محاسبه کرده و حاصل را p می‌نامیم. بلاک p ام زنجیره را برای محاسبات بعدی انتخاب می‌کنیم.

ب) باقیمانده‌ی مبلغ تراکنش را بر تعداد تراکنش‌های بلاک p (در مرحله قبل انتخاب شد) محاسبه کرده و مقدار حاصل را q می‌نامیم. تراکنش q ام بلاک p را انتخاب کرده و با عنوان kk در محاسبات بعدی استفاده خواهیم کرد.

توجه: به این ترتیب به ازای هر تراکنش k در بلاک آخر، تراکنش دیگری (kk) در مجموعه‌ی کل بلاک‌ها انتخاب می‌شود.

پ) تراکنش kk را با یک رشته ۶ بایتی مدل می‌کنیم:

شناسه فرستنده، ۱ بایت، شناسه گیرنده، ۱ بایت، و مبلغ تراکنش ۴ بایت. با کنار هم قرار دادن مقادیر فوق (از چپ به راست)، رشته مورد نظر بدست می‌آید. از بیت شماره 41 % ((n+1) * (k+1)) این رشته شروع کرده و ۸ بیت جدا می‌کنیم تا اوکتت متناظر با تراکنش k ام بدست می‌آید.

به عنوان نمونه تصویر زیر، بلاکی را نشان می‌دهد که چهار تراکنش دارد. اوکتت متناظر با هر یک از این تراکنش‌ها نیز محاسبه شده و از O1 تا O4 نام‌گذاری شده‌اند. R0 نیز یک عبارت ۸ بیتی مجهول است که می‌تواند مقادیر بین صفر تا ۲۵۵ را به خود اختصاص دهد. هدف اصلی این بخش، محاسبه مطلوب‌ترین مقدار R0 است. برای بدست آوردن این

		R0	
Transaction 1	=>	O1	R1 = Lookup(O1 ^ R0)
Transaction 2	=>	O2	R2 = Lookup(O2 ^ R1)
Transaction 3	=>	O3	R3 = Lookup(O3 ^ R2)
Transaction 4	=>	O4	R4 = Lookup(O4 ^ R3)
			R4: Final Result for a particular value of **R0**

[5] Octet

مقدار، فرآیند merge را با در نظر گرفتن تمامی مقادیر ممکن R0 (۲۵۶ حالت) تکرار می‌کنیم و خروجی‌های بدست آمده را مقایسه می‌کنیم. R0 ای که منجر به ایجاد بزرگترین خروجی (از نظر الفبایی) شود را به عنوان حالت مطلوب انتخاب کرده و در خروجی استاندارد چاپ می‌کنیم.

جزئیات فرآیند merge

مقادیر R0 و O1 را تحت عملگر منطقی xor قرار می‌دهیم. مقدار حاصل یک رشته‌ی هشت بیتی است. چهار بیت پرارزش و کم‌ارزش آن را به ترتیب i و j می‌نامیم، خانه متناظر با سطر i و ستون j را از جدول زیر انتخاب کرده و مقدار این خانه را R1 می‌نامیم. فرآیند merge را با R1 و O2 ، R2 و O3 و ... تکرار می‌کنیم تا آخرین R (در مثال بالا R4) بدست آید. R4 خروجی مورد نظر ما به ازای یک مقدار خاص R0 است.

0x0C	0xB4	0x64	0xB0	0xC1	0x18	0x87	0x09	0x76	0x50	0x29	0x3E	0x34	0xE9	0x7B	0x75
0x9A	0x23	0xCC	0x24	0x5F	0x9C	0xE2	0x61	0x77	0xA6	0x35	0x12	0x98	0xA2	0xA4	0xE8
0xDE	0x02	0xE3	0xEB	0x47	0xE4	0x32	0x33	0x19	0x6C	0x06	0xD9	0xDA	0xF0	0xC6	0x6D
0x84	0x37	0xAD	0xA1	0x48	0xB6	0xDD	0xB2	0x89	0x1E	0x7D	0x58	0xE7	0xB5	0x9B	0xFA
0xC7	0x17	0xDB	0xF5	0xF9	0x43	0xEF	0x71	0xC8	0x67	0xF7	0x27	0x5D	0x01	0x9F	0xD0
0xAE	0xDF	0x66	0x7A	0x99	0x21	0x94	0x7E	0x0B	0x68	0x53	0x5E	0x3A	0x4A	0x8E	0x57
0x9D	0x8B	0x72	0xE6	0x3C	0xAA	0x60	0x97	0xC3	0x00	0xDC	0xA8	0x39	0xBB	0x6A	0x38
0xD7	0x11	0xFB	0xE0	0x7F	0xFE	0xEC	0xD4	0xD2	0xCD	0x70	0x78	0x3D	0x0E	0x14	0xF6
0x8F	0x69	0xBD	0x3B	0x45	0x90	0x93	0xEA	0xF3	0x4C	0x4D	0xD8	0x2D	0xD1	0x1F	0xF1
0xB3	0x2A	0xD6	0x86	0x07	0x22	0xD5	0x46	0xAC	0xE5	0x0F	0x96	0x74	0x3F	0xD3	0x80
0x26	0xCB	0x54	0x73	0xB9	0x44	0x0D	0xC0	0x41	0x8A	0xAB	0x15	0x2E	0x83	0x08	0xED
0xCE	0x16	0x6B	0x03	0xC5	0x52	0xFD	0x81	0xAF	0x8D	0xA0	0xBF	0xF2	0x10	0x5A	0x4E
0xA3	0x4F	0xFC	0x88	0x9E	0x25	0x95	0x5C	0xA5	0x92	0x8C	0x51	0xBA	0x56	0x6E	0x31
0x40	0x1B	0x91	0x85	0x7C	0x49	0x79	0xC2	0x2B	0xBC	0x42	0x6F	0x28	0xF8	0xFF	0xCF
0x59	0x63	0x1A	0xC9	0xBE	0xE1	0xC4	0x13	0xF4	0x55	0xEE	0x65	0xB8	0xB7	0x20	0xA7
0x30	0x2F	0x4B	0xA9	0x04	0x05	0xCA	0x2C	0x0A	0x36	0x62	0x5B	0x1D	0x1C	0xB1	0x82

ورودی و خروجی نمونه

جدول زیر دستورهای تعریف‌شده در سامانه و عملکر مطلوب هر یک را به صورت خلاصه مشخص کرده‌است.

`process_transaction from to amount timestamp`	پردازش یک تراکنش
`end_block`	پایان یک بلاک
`print_accounts`	چاپ حساب‌های کاربری در خروجی استاندارد
`print_block`	چاپ تراکنش‌های بلاک فعلی

نتیجه هر یک از این دستورها در خروجی استاندارد چاپ خواهد شد. توجه کنید که رعایت فرمت مطرح شده الزامی است و در غیر این صورت ممکن است نمره مربوط به موارد آزمون را به صورت کلی از دست بدهید.

در صورتی که رسیدگی به درخواست اپراتور با موفقیت خاتمه یافت، نیازی به چاپ پیام موفقیت نیست، مگر اینکه در صورت تمرین از شما خواسته باشیم. اگر هنگام رسیدگی به یک درخواست با خطا روبرو شدید، پیام خطای مناسب را در ۱ خط با فرمت "ERR: message" در خروجی استاندارد چاپ کنید. message متن پیام خطا است که مقدار آن به دلخواه توسط شما مشخص می‌شود.

Sample Input

```
3
0,sepehr ,1000
1 , agha pesar , 200
2, gol pesar ,   100
print_accounts
process_transaction 0 1 25 1454946270
process_transaction 2 1 30 1454948745
print_block
end_block
process_transaction 0 1 10 1454949270
process_transaction 0 2 200 1454951105
process_transaction 1 2 60 1454953210
print_block
end_block
print_accounts
process_transaction 0 1 20 1454955210
process_transaction 0 2 20 1454957210
end_block
process_transaction 0 2 1000 1454957210
```

Sample Output

```
UID         Name                            Balance
-------------------------------------------------
0           sepehr                             1000
1           agha pesar                          200
2           gol pesar                           100
Block    :0        Transaction Count    :2        Last Timestamp     :1454948745
--------------------------------------------------------------------------------
1454946270      sepehr                    agha pesar                          25
1454948745      gol pesar                 agha pesar                          30
Best hash achievable when R0 = 36
Block    :1        Transaction Count    :3        Last Timestamp     :1454953210
--------------------------------------------------------------------------------
1454949270      sepehr                    agha pesar                          10
1454951105      sepehr                    gol pesar                          200
1454953210      agha pesar                gol pesar                           60
Best hash achievable when R0 = 238
UID         Name                            Balance
-------------------------------------------------
0           sepehr                             765
1           agha pesar                         205
2           gol pesar                          330
Best hash achievable when R0 = 32
ERR: insufficient fund
```

نحوه‌ی تحویل

فایل برنامه‌ی خود را با نام **A1-SID.cpp** در سایت درس بارگذاری کنید. (SID پنج رقم آخر شماره‌ی دانشجویی شماست. به عنوان مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۹۴۱۲۳ است، نام فایل شما باید A1-94123.cpp باشد.)

دقت کنید

- برنامه‌ی شما باید در سیستم‌عامل لینوکس نوشته شده و با مترجم g++ کامپایل شود.
- در چاپ کردن خروجی نهایت دقت را به خرج دهید.
- به فرمت و نام فایل‌های خود دقت کنید. در صورتی که هر یک از موارد گفته شده رعایت نشود، نمره‌ی صفر برای شما در نظر گرفته می‌شود.
- در صورت کشف تقلب در کل و یا قسمتی از تمرین، برای هر دو طرف نمره‌ی ۱۰۰ ـ منظور خواهد شد.