استاد: رامتین خسروی
طراح: فرزاد حبیبی
مهلت تحویل: جمعه ۱۹ مهر ۱۳۹۸، ساعت ۲۳:۵۵

۱ پَدارص [۱]

هدف از این تمرین آشنایی شما با ورودی/خروجی استاندارد [۲] ، کار با فایل و همین‌طور بردار [۳] ها در زبان C++ می‌باشد. شما در این تمرین به پیاده‌سازی یک پیام‌رسان سادهٔ داخلی می‌پردازید. این پیام‌رسان روی یک سیستم اجرا می‌شود و توانایی نوشتن پیام‌ها در فایل متناظر با افراد را دارد.

داستان از این قرار است که آزمایشگاه روش‌های صوری [۴] و مهندسی نرم‌افزار دانشگاه تهران، یک کامپیوتر شخصی [۵] بیشتر در اختیار ندارد. دانشجویان و اساتید در این آزمایشگاه، کارهای خودشان را با این کامپیوتر و زمانی که کسی با آن کاری نداشته باشد، انجام می‌دهند. هر کدام از دانشجویان و اساتید در زمان انجام کارهایشان ممکن است بخواهند به شخص دیگری در این آزمایشگاه نکته‌ای را یادآوری کنند، اما تضمینی نیست که در آن زمان در آزمایشگاه حضور داشته باشند؛ لذا، آن‌ها از شما می‌خواهند برنامه‌ای بنویسید که با استفاده از آن بتوانند به همکارانشان پیام بدهند و هر فرد بتواند پیام‌هایش را در سیستم ببیند.

۱.۱ دریافت اطلاعات افراد

زمانی که برنامه شروع به اجرا شدن می‌کند، اطلاعات افراد درون آزمایشگاه به همراه رابطه‌شان با یکدیگر از یک فایل به اسم poeple.txt که در کنار فایل اجرایی و در پوشهٔ [۶] آن قرار دارد، خوانده می‌شود. هر خط از این فایل دارای قالبی مشابه قالب زیر می‌باشد.

$$ID\ A_0\ A_1\ \dots\ A_n$$

- در هر خط از این فایل اطلاعات یک فرد آمده‌است.
- اعداد در هر خط با یک فاصله [۷] از یکدیگر جدا شده‌اند.
- اولین عدد هر خط شمارهٔ شناسایی [۸] یک فرد می‌باشد.
- اعداد بعدی در هر خط شمارهٔ شناسایی افرادی هستند که با او ارتباط دارند و می‌توانند به آن‌ها پیام ارسال یا دریافت کند. این رابطه یک رابطه‌ی دوطرفه می‌باشد و آن‌ها نیز می‌توانند به این شخص پیام ارسال کنند.

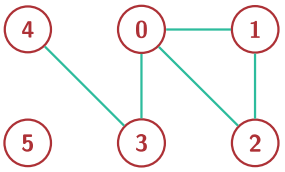
برای مثال به نمونه‌ی زیر توجه کنید.

```
0 1 2 3
1 2
2
3 4
4
5
```

[۱] پیام‌رسان داخلی آزمایشگاه روش‌های صوری

[2] standard I/O
[3] vector
[4] formal methods
[5] PC
[6] directory
[7] space
[8] ID

افراد متناظر با این متن را می‌توان به شکل زیر رسم کرد. افرادی که در شکل زیر به یک‌دیگر وصل هستند می‌توانند به یک‌دیگر پیام بدهند. مثلاً شخصی با شمارهٔ شناسایی 0 می‌تواند به افرادی با شمارهٔ شناسایی 1,2,3 پیام ارسال کند؛ همین‌طور، شخصی با شمارهٔ شناسایی 4 می‌تواند به فردی با شمارهٔ شناسایی 3 پیام ارسال کند. یا اینکه فرد شمارهٔ 5 توانایی ارسال یا دریافت پیام از هیچ‌کس را ندارد.



۲.۱ ارسال پیام

افراد مختلف با استفاده از این برنامه می‌توانند پیام ارسال کنند. بعد از این‌که رابطهٔ افراد در برنامهٔ شما از فایل مربوط به آن خوانده شد، برنامه‌تان باید بتواند از ورودی استاندارد دستور ارسال پیام را دریافت کند. قالب ورودی این دستور به شکل زیر می‌باشد.

senderID receiverID message

- هر خط به ترتیب شامل شمارهٔ شناسایی فرد ارسال کننده، شخصی که قصد ارسال پیام به او را دارد و پیامی که می‌خواهد ارسال کند می‌باشد.
- هر کدام از این سه، با یک فاصله از یکدیگر جدا شده‌اند.
- در صورتی که پیام با موفقیت ارسال شود در خط بعدی sent و در صورتی که این پیام قابل ارسال نباشد عبارت failed چاپ می‌شود.
  - زمانی‌که یک فرد طی اطلاعات اولیه به شخص دیگری دسترسی نداشته باشد، پیام قابل ارسال نیست.
  - در صورتی که هر کدام از شمارهٔ شناسایی‌های وارد شده وجود خارجی نداشتند و در فایل اطلاعات اولیه نیامده باشند، امکان ارسال پیام وجود ندارد.
- خود پیام می‌تواند حاوی کاراکتر فاصله باشد.
- در صورتی که در انتهای پیام فاصلهٔ اضافی وجود داشته باشد، باید این فاصله‌ها را حذف کنید و در فایل پیام‌های دریافتی ذخیره کنید.

برای نمونه در مثال بخش قبل ، فرض کنید که فرد شمارهٔ 0 قصد ارسال پیام به شخص 3 و 5 را دارد. ورودی و خروجی کنسول به شکل زیر می‌باشد.

ورودی نمونه	خروجی نمونه
`0 3 slm. file haye pdf dobare barresi shavad.` `0 5 slm dust e aziz!`	`sent` `failed`

۳.۱ ذخیرهٔ پیام‌ها

به محض اینکه یک پیام ارسال می‌شود، باید در فایل متناظر با دریافت‌کننده نوشته شود. در واقع در کنار فایل اجرایی برنامه و در پوشه‌ی مشابه، به ازای هر فرد که در فایل اطلاعات اولیه آمده‌است یک فایل متنی [۹] وجود دارد و پیام‌های دریافتی در آن ذخیره‌سازی می‌شود.

بنابراین با بسته شدن برنامه پیام‌ها همچنان باقی می‌مانند و هر کس با اجرای دوباره‌ی برنامه قابلیت مشاهده‌ی آن‌ها را دارد. هر کدام از فایل‌های متنی شامل پیام‌های دریافتی شخص می‌باشد که هر خط آن یک پیام را مشخص می‌کند. قالب ذخیره‌سازی پیام‌ها باید به شکل زیر باشد.

senderID message

- شمارهٔ شناسایی فرد ارسال کننده‌ی پیام و خود پیام با یک فاصله از یکدیگر جدا شده‌اند.
- بعد از آخرین پیام در این فایل کاراکتر \n وجود دارد.
- در انتهای پیام‌ها هیچ کاراکتر فاصله‌ای وجود ندارد و باید همانطور که در بخش ارسال پیام گفته شد، حذف شده باشند.

به عنوان نمونه فایل مربوط به پیام‌های دریافتی یک شخص می‌تواند به شکل زیر باشد.

```
0 slm. file haye pdf dobare barresi shavad.
4 farzad yadet bashe pdf haro behet bedam.
4 pdf haro vasat email kardam
0 in che cherto pertie tu pdf ha neveshtin?
0 farda saat 9:30 biaid otaghe man jenabe habibi!

```

.

۴.۱ نمایش پیام‌ها

در این بخش تنها کافیست محتویات درون فایل هر شخص را چاپ کنید. قالب دستور نمایش پیام‌ها به شکل زیر می‌باشد.

show_msg ID

- دستور show_msg و شمارهٔ شناسایی فرد با یک فاصله از یکدیگر جدا شده‌اند.
- در صورتی که شمارهٔ شناسایی فرد وجود نداشته باشد عبارت failed در خروجی استاندارد چاپ می‌شود.

[9] txt

۵.۱ نکات پایانی

- در هر کدام از ورودی‌های کاربر، در صورتی که دستور ورودی وجود نداشت و جز هیچ‌کدام از دستورهای ذکر شده [۱۰] نبود باید عبارت failed در خروجی استاندارد چاپ شود.
- قالب فایل اطلاعات اولیه همواره همچون قالب ذکر شده می‌باشد و از شما انتظار نمی‌رود که هر گونه خطایی در آن را تشخیص دهید.
- برای خواندن یک خط کامل از ورودی می‌توانید از تابع `getline` در سرآیند `iostream` [۱۱] کمک بگیرید.
- برنامه همواره در حال اجرا شدن و دستور گرفتن می‌باشد و تا زمانی که کارکتر EOF دریافت نشود به اجرای خود ادامه می‌دهد.
- برای آشنایی با خواندن از فایل و نوشتن در آن می‌توانید به این لینک مراجعه کنید.

۶.۱ ورودی و خروجی نمونه

به ورودی و خروجی نمونهٔ زیر که با فرض اطلاعات اولیه‌ی بخش ۱.۱ آمده‌است توجه کنید.

ورودی نمونه	خروجی نمونه
0 3 slm. file haye pdf dobare barresi shavad. 0 5 slm dust e aziz! 8 5 haletun khube? show_ms 0 4 3 farzad yadet bashe pdf haro behet bedam. 4 3 pdf haro vasat email kardam 0 3 in che cherto pertie tu pdf ha neveshtin? 0 3 farda saat 9:30 biaid otaghe man jenabe habibi! show_msg show_msg 3 3 0 chashm hatman.	sent failed failed failed sent sent sent sent failed 0 slm. file haye pdf dobare barresi shavad. 4 farzad yadet bashe pdf haro behet bedam. 4 pdf haro vasat email kardam 0 in che cherto pertie tu pdf ha neveshtin? 0 farda saat 9:30 biaid otaghe man jenabe habibi! sent

- در ورودی خط ۲ امکان دسترسی از فرد 0 به شخص 5 وجود ندارد و عبارت failed چاپ می‌شود.
- در خط‌های ۴و ۹ دستورات به اشتباه وارد شده‌اند. بنابراین عبارت failed چاپ شده‌است.
- خط ۱۰ پیام‌های فردی با شماره‌شناسایی 3 را نمایش می‌دهد.
- در خط ۳ شخصی با شمارهٔ شناسایی 8 وجود ندارد و عبارت failed چاپ شده‌است.

[۱۰] ارسال پیام و نمایش پیام‌ها

[11] header

۲ نحوهٔ تحویل

برنامهٔ خود را با نام A1-SID.cpp در صفحهٔ CECM درس بارگذاری کنید که SID شمارهٔ دانشجویی شماست؛ برای مثال اگر شمارهٔ دانشجویی شما ۸۱۰۱۹۷۹۹۹ باشد، نام پروندهٔ شما باید A1-810197999.cpp باشد.

- برنامهٔ شما باید در سیستم‌عامل لینوکس و با مترجم g++ با استاندارد c++11 ترجمه و در زمان معقول برای ورودی‌های آزمون اجرا شود.
- از صحت قالب[۱۲] ورودی‌ها و خروجی‌های برنامهٔ خود مطمئن شوید. برنامهٔ شما در هنگام تحویل حضوری به صورت اتوماتیک تست می‌شود؛ لذا، از دادن خروجی‌هایی که در صورت پروژه گفته نشده‌است اجتناب کنید.
- رعایت سبک برنامه‌نویسی درست و تمیز بودن برنامه‌ی شما در نمرهٔ تمرین تأثیر زیادی دارد.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

آ مقایسهٔ خروجی برنامه با خروجی مورد انتظار

مقایسهٔ خروجی برنامه با خروجی مورد انتظار با چشم شاید برای برنامه‌های کوچک که خروجی کمی تولید می‌کنند و روند اجرای کوتاهی دارند میسر باشد، برای برنامه‌های بزرگ‌تر با مسیر اجرای پیچیده کاری دشوار است. برای این کار می‌توان از ابزارهایی که در سیستم‌عامل لینوکس در دسترس است استفاده کرد.

در حالت عادی، برای ترجمه و اجرای یک برنامه از این دستورها استفاده می‌شود:

```
g++ -std=c++11 helloworld.cpp
./a.out
```

در این حالت برنامه ورودی‌اش را از ورودی استاندارد stdin (خط فرمان) می‌خواند و خروجی را نیز در خروجی استاندارد stdout (صفحه‌ی خط فرمان) می‌نویسد.

برای اجرای راحت‌تر برنامه، می‌توان ورودی را در پرونده مانند in.txt نوشت و سپس محتوای آن را به ورودی استاندارد تغییر مسیر[۱۳] داد تا هنگام اجرای مکرر برنامه نیازی به نوشتن مکرر ورودی‌های مختلف در خط فرمان نباشد:

```
./a.out < in.txt
```

همچنین، می‌توان خروجی برنامه را به پرونده‌ای مانند out.txt تغییر مسیر داد تا بتوان بعداً هم به آن دسترسی داشت:

```
./a.out > out.txt
```

ترکیب این دو عمل نیز امکان‌پذیر است:

```
./a.out < in.txt > out.txt
```

[12] format
[13] redirect

فرض کنیم خروجی مورد انتظار برای ورودی in.txt در پرونده‌ای به نام sol.txt قرار دارد. می‌توان با استفاده از دستور diff خروجی حاصل از اجرای برنامه را با خروجی مورد انتظار مقایسه کرد.

برای این کار، ابتدا ورودی را از in.txt به برنامه می‌دهیم و خروجی برنامه را در پرونده‌ای مانند out.txt ذخیره می‌کنیم. سپس با دستور diff پرونده‌ی out.txt را با sol.txt مقایسه می‌کنیم.

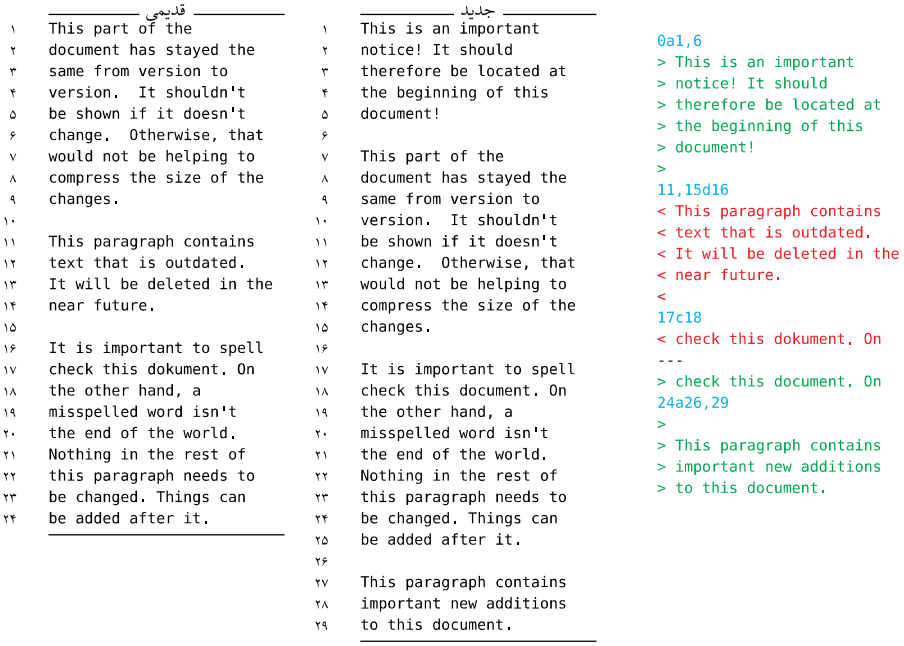
```
g++ -std=c++11 helloworld.cpp
./a.out < in.txt > out.txt
diff out.txt sol.txt
```

اگر پرونده‌ها یکسان باشند، دستور diff هیچ خروجی‌ای تولید نمی‌کند. وگرنه، تفاوت‌های دو پرونده را نشان می‌دهد.

هر بخش از خروجی این دستور با شماره‌ی خطوط آغاز می‌شود: شماره‌ی خطوط در پرونده‌ی قدیمی (سمت چپ)، یکی از حروف d، a یا c و شماره‌ی خطوط در پرونده‌ی جدید (سمت راست). حرف میان شماره‌ی خطوط نوع تغییرات را نشان می‌دهد:

- **d: حذف شدن** محتوای محذوف بعد از < نمایش داده می‌شود.
- **a: افزوده شدن** محتوای جدید بعد از > نمایش داده می‌شود.
- **c: تغییر** محتوای قدیمی بعد از < نمایش داده می‌شود. سپس خطی شامل --- می‌آید. بعد از آن، محتوای جدید بعد از > نمایش داده می‌شود.

به این مثال[14] توجه کنید:



[14] https://en.wikipedia.org/wiki/Diff