استاد: **رامتین خسروی** طراحان: امیرحسین حبیب‌وند، ژیوار صورتی، غزل کلهر

مهلت تحویل: جمعه ۱۶ اسفند ۱۳۹۸، ساعت ۲۳:۵۵

برنامه پیشنهاد انتخاب واحد

یکی از دغدغه‌های اصلی برای هر دانشجوی کارشناسی، تهیه‌ی یک برنامه معقول برای اخذ واحد در ترم‌های آینده است. در این تمرین قصد داریم با نوشتن یک برنامه، این فرآیند را برای دانشجویان تسهیل کنیم.

هدف این تمرین، آشنایی شما با طراحی بالا به پایین[1] یک مسئله و کار با پرونده[2]ها است. در این راستا تمرین را به ۳ مرحله تقسیم کرده‌ایم تا فرآیند طراحی و پیاده سازی را به صورت مرحله به مرحله انجام دهید.

توصیه می‌کنیم ابتدا کل تمرین را مطالعه کرده و سپس شروع به پیاده‌سازی موارد خواسته شده کنید. پیاده‌سازی هر مرحله به طور جداگانه ارزیابی خواهد شد.

اطلاعات ورودی برنامه

ورودی‌های برنامه شامل اطلاعات درس‌ها و نمره‌های دانشجویان است. این دو مجموعه اطلاعات در دو پرونده ذخیره شده‌اند که آدرس آنها در خط فرمان به برنامه داده می‌شود. لازم است برنامه شما در ابتدای اجرا محتویات این دو پرونده را بخواند و اجزای آن را تجزیه و در متغیرهای مناسب نگهداری کند.

به عنوان مثال، اگر پرونده اجرایی برنامه شما بعد از ترجمه a.out باشد، چنین دستوری در خط فرمان اجرا خواهد شد:

```
./a.out ./courses.csv ./grades.csv
```

برای این که با خواندن آرگومان‌های خط فرمان آشنا شوید پیوست ۱ را مطالعه نمایید.

قالب این دو پرونده، مقادیر جداشده با کاما (Comma-Separated Values) یا CSV است. در این نوع پرونده، هر سطر دارای تعدادی قلم داده است که با کاراکتر کاما (,) از هم جدا شده‌اند. در سطر اول این پرونده‌ها عنوان قلم‌داده‌ها با کاما از یکدیگر جدا می‌شوند.

به عنوان مثال دو سطر اول پرونده درس‌ها چنین شکلی دارد (ساختار این پرونده جلوتر توضیح داده خواهد شد):

[1] Top-Down Design

[2] File

```
Id,Name,Units,Schedule,Prerequisites
1,ICSP,4,Sat-09:00-10:30/Mon-09:00-10:30/Thu-09:00-10:30,0
2,AdvProg,3,Sun-10:30-12:00/Tue-10:30-12:00,1
```

این اطلاعات متناظر با جدولی به شکل زیر هستند:

Id	Name	Units	Schedule	Prerequisites
1	ICSP	4	Sat-09:00-10:30/Mon-09:00-10:30/Thu-09:00-10:30	0
2	AdvProg	3	Sun-10:30-12:00/Tue-10:30-12:00	1

ساختار پرونده‌های ورودی

۱. اطلاعات درس‌ها

آرگومان اول خط فرمان برنامه، مسیر پرونده اطلاعات درس‌ها را مشخص می‌کند. هر سطر پرونده مربوط به اطلاعات درس‌های دانشکده نشان‌دهنده یک درس است که دست کم یک جلسه در هفته تشکیل می‌شود. هر سطر این پرونده شامل تعدادی ستون به شرح زیر است:

نام ستون	مقدار نمونه	شرح ستون
Id	8101119	شناسه یکتای درس (یک عدد طبیعی)
Name	AdvancedProgramming	نام درس (یک رشته شامل تعدادی حرف بزرگ و کوچک و عدد)
Units	3	تعداد واحد درس (یک عدد طبیعی)
Schedule	Sun-10:30-12:00/Tue-10:30-12:00	زمان‌های ارائه کلاس‌های درس در قالب DoW-HH:MM-HH:MM/.../DoW-HH:MM-HH:MM که در آن DoW یکی از مقادیر Sat, Sun, Mon, Tue, Wed, Thu, Fri است و HH:MM نشان‌دهنده زمان شروع یا پایان است.
Prerequisites	8101123-8101001	شناسه درس‌های پیش‌نیاز این درس که با خط تیره (-) از یکدیگر جدا شده‌اند. در صورتی که درسی هیچ پیش‌نیازی نداشته باشد یک عدد 0 در این فیلد قرار می‌گیرد.

در خواندن ورودی می‌توانید فرض کنید که

- مقادیر همه ستون‌ها معتبر هستند،
- شناسه هیچ ۲ درسی برابر نیست،

- در گراف پیش‌نیازی درس‌ها دور وجود ندارد (یعنی هیچ درسی مستقیم یا با واسطه پیش‌نیاز خودش نیست)،
- درس‌هایی که به عنوان پیش‌نیاز معرفی می‌شوند در جدول درس‌ها وجود دارند و قبل از این درس آمده‌اند.

۲. نمره‌های دانشجو

هر سطر پرونده نمره‌ها حاوی نمره کسب شده توسط دانشجو در یک درس است. هر سطر شامل ۲ ستون است که ستون اول (با سر ستون Id) شناسه درس و ستون دوم نمره (با سر ستون Grade) کسب شده توسط دانشجو را نشان می‌دهد که یک عدد اعشاری بین صفر و بیست است. دانشجو برای گذراندن درس لازم است نمره بزرگ‌تر یا مساوی ۱۰ در آن درس کسب کند. اگر دانشجو درسی را گذرانده باشد نمی‌تواند دوباره آن را اخذ کند. اگر دانشجو یک درس را بیشتر از یکبار اخذ کرده باشد، در پرونده نمره‌ها بیشتر از ۱ سطر مربوط به آن درس وجود خواهد داشت.

تضمین می‌شود که نمره قبولی کسب شده در یک درس در پرونده در سطری پایین‌تر از نمره‌های قبلی فرد در آن درس باشد. همچنین شناسه دروس معرفی شده در این پرونده حتما در پرونده مربوط به درس‌ها هم وجود دارند.

مراحل برنامه

شما باید برنامه مورد نظر را در ۳ مرحله پیاده کنید. توجه کنید که در نهایت باید هر مرحله را در یک پرونده جداگانه در سامانه بارگذاری کنید. قبل از بارگذاری پاسخ تمرین حتما بخش نحوه تحویل را مطالعه کنید.

مرحله ۱ - تعیین درس‌های قابل اخذ

در این مسئله باید برنامه‌ای بنویسید که آدرس دو پرونده توضیح‌شده در بخش قبل را از خط فرمان بگیرد و همه درس‌های قابل اخذ دانشجو را پیدا کند. دانشجو امکان اخذ درسی را دارد که قبلا آن را نگذرانده باشد و همه درس‌های پیش‌نیاز آن را گذرانده باشد. به عنوان خروجی این مرحله کافیست شناسه درس‌های قابل اخذ را در خروجی استاندارد چاپ کنید. شناسه درس‌ها را بر اساس نام درس‌ها و به صورت صعودی مرتب و هر کدام را در یک خط چاپ کنید.

روش حل مرحله ۱

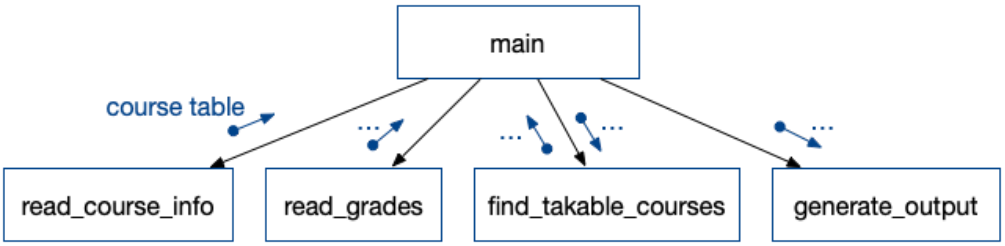
از آنجا که هدف این تمرین طراحی بالا به پایین است، به عنوان الگویی برای مراحل بعد، درباره روش حل مسئله این مرحله توضیحاتی ذکر می‌شود.

قدم اول - در بالاترین سطح، مراحلی که برای حل این مسئله لازم هستند را می‌توان به بخش‌های زیر تقسیم کرد:

1. خواندن اطلاعات درس‌ها
2. خواندن اطلاعات نمره‌ها

3. تعیین درس‌های قابل اخذ
4. تولید خروجی مناسب

برای هر یک از این چهار مرحله می‌توان یک تابع تعریف کرد که در نمودار زیر نمایش داده شده‌اند.



قدم دوم - مرحله بعد مشخص کردن داده‌های مورد تبادل بین توابع است که در شکل فوق برای تابع read_course_info مشخص شده است و برای سایر توابع به شما واگذار شده است. برای هر یک از اقلام داده‌ای، باید ساختار و تایپ‌های لازم به شکل مناسبی تعریف شوند. به عنوان مثال، تایپ CourseTable به عنوان تایپ مقدار بازگشتی read_course_info می‌تواند به شکل زیر تعریف شود:

```
typedef vector<Course> CourseTable;
struct Course {
      int id;
      string name;
      int units;
      ??? schedule;
      ??? prerequisites;
};
```

در تکمیل تعریف تایپ فوق، تایپ‌های مناسبی برای فیلدهای schedule و prerequisites تعریف کنید. برای این کار، دقت داشته باشید که تایپ schedule لیستی از بازه‌های زمانی است که آن هم به نوبه خود ساختار مرکبی دارد. بنابراین در تعریف زمان‌بندی کلاس‌ها تایپ‌های متفاوتی درگیر هستند که آنها را به دقت و به صورت مجزا تعریف کنید.

همچنین، در تعریف تایپ فیلد prerequisites به این موضوع دقت کنید که تعیین یک نام برای این تایپ (به کمک typedef) به استفاده مجدد از آن در بخش دیگری از برنامه کمک خواهد کرد.

قدم سوم - در صورتی که بعضی از توابع هنوز جای تجزیه بیشتر داشته باشند، قدم‌های قبل برای آنها تکرار می‌شود. به عنوان نمونه، به نظر می‌رسد تابع find_takable_courses چنین خاصیتی داشته باشد. اگر مراحل انجام این تابع را به سبک قدم اول به صورت شبه‌کد بنویسیم متوجه می‌شویم پتانسیل تعریف توابع کوچک‌تری وجود دارد. حتی تابع read_course_info نیز چنین پتانسیلی را دارد.

مرحله ۲ - پیشنهاد برنامه ترم

در این مرحله برنامه‌ی شما باید با توجه به معدل کل دانشجو و نیز لیست نمرات، برنامه‌ای پیشنهادی برای اخذ واحد در ترم آینده ارائه دهد. معدل کل دانشجو می تواند عددی اعشاری در بازه ۰ تا ۲۰ باشد که برابر با میانگین وزن‌دار نمرات دانشجو است. در صورتی که معدل کل دانشجو برابر با یا بالاتر از ۱۷ باشد می‌تواند حداکثر ۲۴ واحد و در

غیر این صورت می‌تواند حداکثر ۲۰ واحد اخذ کند. همچنین توجه داشته باشید که حداقل تعداد واحد قابل اخذ در یک ترم، ۱۲ واحد است مگر اینکه تعداد واحد ممکن برای اخذ توسط دانشجو از این مقدار کمتر باشد.

برای پیدا کردن درس‌های ترم آینده دانشجو از الگوریتم زیر استفاده کنید:

1. لیست دروس قابل اخذ توسط دانشجو (که در مرحله قبل به دست آوردید) را به صورت نزولی بر اساس تعداد واحدهای درس و بین دروس با تعداد واحد یکسان بر حسب حروف الفبا و به صورت صعودی مرتب کنید.
2. به ترتیب روی درس‌ها پیمایش می‌کنیم و برای هر درس مرحله ۳ را اجرا می‌کنیم.
3. در صورتی که با انتخاب این درس جمع واحدها بیشتر از تعداد واحد مجاز می‌شود یا این درس با یکی از درس‌هایی که تا به اینجا انتخاب شده‌اند تداخل دارد، درس را جزء درس‌های انتخابی در نظر نمی‌گیریم و به سراغ درس بعدی می‌رویم. در غیر این صورت درس را به دروس انتخابی اضافه می‌کنیم.

منظور از تداخل، وجود هم‌پوشانی بین حداقل دو کلاس از دو درس است. دقت کنید اگر یکی از زمان‌های ابتدا و انتهای بازه‌ها با یکدیگر یکی باشند تداخل محسوب نمی‌شود. مثلاً کلاس ۷:۳۰ تا ۹:۰۰ با کلاس ۹:۰۰ تا ۱۰:۳۰ در همان روز تداخل ندارد.

شناسه دروس خروجی الگوریتم بالا را در خروجی استاندارد چاپ کنید. شناسه‌ها را بر اساس نام درس‌ها و به صورت صعودی مرتب و هر کدام را در یک خط چاپ کنید.

مرحله ۳ - پیشنهاد برنامه تحصیل (امتیازی)

در این مرحله انتظار می‌رود که برنامه شما، برنامه ترم‌های آتی دانشجو تا زمانی که همه دروس را بگذارند را تولید کند. برای این منظور فرض کنید که دانشجو همه درس‌هایی که اخذ می‌کند را پاس می‌کند و همچنین معدل هر ترم او ۵ درصد بیشتر از معدل کلش تا آن ترم می‌شود. برای پیدا کردن برنامه هر ترم از الگوریتم مرحله ۲ استفاده کنید. شناسه دروسی که دانشجو در هر ترم باید اخذ کند را بر اساس نام دروس و به صورت صعودی مرتب و در پرونده‌هایی با نام‌های semester1.sched و semester2.sched و ... در کنار پرونده اجرایی ذخیره کنید.

نکات پایانی

- تعریف دقیق ساختار تایپ‌های مربوط به داده‌های مسئله و نام‌گذاری مناسب برای آنها و همچنین تعریف سلسله‌مراتبی از توابع که در نهایت به توابع کوچکی که هر کدام یک مسئولیت دارند جزء مهمی از ارزشیابی این تمرین را به خود اختصاص می‌دهد.
- راه‌حل گفته شده برای مرحله اول لازم نیست که دقیقا به همین صورت پیاده شود، اما انتظار می‌رود که مساله به همین صورت به زیرمساله‌های کوچک‌تر تقسیم شود و این روند برای زیرمساله‌ها نیز تکرار شود.
- نام پرونده حاوی لیست درس‌ها در ورودی اول خط فرمان به شما داده می‌شود. تضمین می‌شود که این پرونده وجود داشته باشد و همچنین فرمت آن به صورت گفته شده باشد.

- نام پرونده حاوی نمراتی که دانشجو در هر درس گرفته در ورودی دوم خط فرمان به شما داده می‌شود. تضمین می‌شود که این پرونده وجود داشته باشد و همچنین فرمت آن به صورت گفته شده باشد.
- با توجه به این‌که تجزیه پرونده‌های CSV بخشی از تمرین است، استفاده از کتابخانه‌های موجود جهت تجزیه کردن این پرونده‌ها، **قابل قبول نیست**.
- برای آشنایی بیشتر با ورودی و خروجی‌های برنامه حتما به نمونه ورودی و خروجی که در کنار تمرین قرار گرفته است، مراجعه کنید.
- قبل از شروع پیاده‌سازی به طراحی برنامه خود زمان کافی اختصاص دهید و به جوانب مختلف آن فکر کنید.
- برای ذخیره‌سازی موارد مختلف می‌توانید (هر چند الزامی ندارید) از داده‌ساختار map استفاده کنید. برای آشنایی با داده‌ساختار map می‌توانید به اینجا و اینجا مراجعه کنید.

نحوهٔ تحویل

کد مربوط به هر مرحله را در ۳ پرونده با نام‌های *1.cpp* ، *2.cpp* و *3.cpp* ذخیره کنید. سپس همه این پرونده‌ها را در یک پرونده فشرده از نوع زیپ با نام A3-SID.zip در صفحهٔ CECM درس بارگذاری کنید که SID شمارهٔ دانشجویی شماست؛ برای مثال اگر شمارهٔ دانشجویی شما ۸۱۰۱۹۸۹۹۹ باشد، نام پروندهٔ شما باید A3-810198999.zip باشد.

- برای فشرده‌سازی حتماً از zip استفاده کنید و استفاده از فرمت‌های دیگر نظیر tar، rar و ... غیرقابل قبول است.
- 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 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.

پیوست ۱ - آرگومان‌های خط فرمان

آرگومان‌های خط فرمان آرگومان‌هایی هستند که سیستم‌عامل در زمان اجرای برنامه آن‌ها را به برنامه انتقال می‌دهد. برنامه می‌تواند آن‌ها را نادیده بگیرد یا از آن‌ها استفاده کند.

برای استفاده از این آرگومان‌ها، تابع main باید به صورت زیر نوشته شود:

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

دو آرگومان تابع را می‌توان برای دسترسی به آرگومان‌های خط فرمان استفاده کرد:

- argc عدد صحیح؛ تعداد آرگومان‌های خط فرمان داده شده به برنامه

این مقدار حداقل برابر با یک است؛ زیرا دستور اجرای برنامه (نام پرونده اجرایی) حتماً در زمان اجرای برنامه مورد استفاده قرار می‌گیرد و همواره به‌عنوان آرگومان‌های خط فرمان شماره صفر به برنامه داده می‌شود.

- argv آرایه‌ای از رشته‌های مدل زبان C؛ آرگومان‌های خط فرمان داده شده به برنامه

به عنوان یک مثال ساده برنامه زیر را در نظر بگیرید:

```
#include <iostream>

int main(int argc, char *argv[])
{
   std::cout << "There are " << argc << " arguments:" << std::endl;

   // Loop through each argument and print its number and value
   for (int count=0; count < argc; ++count)
      std::cout << count << " " << argv[count] << std::endl;

   return 0;
}
```

اگر برنامه به شکل

```
./a.out Myfile.txt 100
```

اجرا شود، خروجی زیر تولید می‌شود:

```
There are 3 arguments:
0 ./a.out
1 Myfile.txt
2 100
```

برای آشنایی بیشتر با نحوه کار آرگومان‌های خط فرمان می‌توانید به این لینک مراجعه کنید.