معادله‌ی ریاضی

- محدودیت زمان: ۵ ثانیه

شلدون که متوجه می‌شود باید با لئونارد هم‌اتاقی شود، قرارداد مخصوص خود را با بندهایی عجیب و غریب آماده می‌کند. او برای این‌که مطمئن شود از این پس در آرامش به سر خواهد برد، قرارداد را نزد لئونارد می‌برد و از او می‌خواهد پس از خواندن قرارداد، امضایش کند و به آن پایبند باشد.

لئونارد پس از خواندن قرارداد از سخت‌گیری‌های شلدون تعجب می‌کند، اما مشکلی با قرارداد ندارد. لئونارد صرفاً برای این‌که شلدون را کمی اذیت کند، به او می‌گوید فقط در صورتی قرارداد را امضا می‌کند که شلدون تسک فعلی لئونارد را برایش انجام دهد. تسک لئونارد این‌گونه است که باید فایلی را بخواند و محاسبات درون آن را انجام دهد؛ سپس با حذف اسپیس‌های اضافه و با نوشتن حاصل محاسبات، آن فایل را ویرایش کند.

در این سوال ابتدا به شما فایلی با نام `input.txt` داده می‌شود که درون آن عبارتی با عملیات ریاضی شامل +، −، * و / وجود دارد. شما باید نتیجه‌ی عملیات گفته شده را به‌دست آورید و عبارت نهایی (که شامل عبارت اولیه با حذف اسپیس‌های اضافی به‌همراه یک علامت = و پاسخ عبارت در انتها است) را داخل فایل `output.txt` نوشته و ذخیره کنید. (بدیهی است که باید اولویت عملگرها را در Evaluate کردن عبارت رعایت کنید.)

همچنین در صورتی که عبارت نوشته شده، جواب نداشته باشد، باید به‌جای نتیجه‌ی عبارت، عبارت `Error` را بنویسید.

ورودی

ورودی برنامه (که از فایل `input.txt` خوانده می‌شود)، تنها شامل یک خط است که در آن تعدادی عدد با علامت‌های ریاضی با فاصله از هم آمده است.

خروجی

خروجی برنامه‌ی شما (که باید آن را داخل فایل `output.txt` بنویسید) باید شامل ۱ خط باشد که در ادامه‌ی اعداد و علامت‌های ریاضیاتی، نتیجه‌ی آن‌ها بدون اسپیس اضافه با یک علامت = نوشته شده باشد.

مثال‌ها

ورودی نمونه ۱

محتوای فایل `input.txt`

```
3 *    5 /2
```

محتوای فایل `output.txt` پس از اجرای برنامه

```
3*5/2=7
```

ورودی نمونه ۲

محتوای فایل `input.txt`

```
8*    -3 + 2/0
```

محتوای فایل `output.txt` پس از اجرای برنامه

```
8*-3+2/0=Error
```

این عبارت به دلیل وجود نداشتن جواب $0 \div 2$ پاسخی ندارد و باید $Error$ بنویسیم.

تست و ارسال

شما می‌توانید تمپلیت شامل چند تست‌کیس و جاج لوکال را از این‌جا دریافت کنید.

جاج لوکال صرفاً برای تست خودتان است و در نهایت باید فایل main.c خود را در کوئرا آپلود کنید و نمره کوئرا ملاک خواهد بود.

معمای دسته‌بندی بزرگ

- محدودیت زمان: ۵ ثانیه

یک شب دیگر در **آپارتمان 4A**، شلدون به تخته‌ی پر از معادلات نگاه می‌کرد و لئونارد مشغول کار بی‌هدف با گوشی‌اش بودکه یک‌دفعه پنی در را باز کرد درحالی‌که یک برگه‌ی مچاله‌شده در دست داشت.

پنی: *«خب، بچه‌ها، کمک لازم دارم. بچه‌ی دوستم تکلیف داره که باید کلمات رو دسته‌بندی کنه و راستش سخت‌تر از چیزی بود که فکر می‌کردم!»*

شلدون (بدون این‌که نگاه کنه): *«پنی، دسته‌بندی کلمات به همون سادگی جدول تناوبیه. حتی تو هم باید راحت از پسش بر بیای. مثلاً "هیدروژن" یه نافلز توی گروه ۱ و دوره‌ی ۱ است.»*

پنی: *«آره ولی کلمات اتم نیستن شلدون. کلماتی مثل "سیب"، "گربه" و "هواپیما".»*

لئونارد: *«فکر کنم منظورش دسته‌بندی‌هایی مثل میوه‌ها، حیوانات و... باشه.»*

شلدون (با هیجان): *«آه! یه مسئله‌ی دسته‌بندی! توی قلمروی تخصص بی‌نهایت من قدم گذاشتی. برید کنار!»*

راج و هاوارد با غذا وارد آپارتمان می‌شوند.

راج: *«چی شده؟ باز شلدون قاطی کرده؟»*

هاوارد: *«به‌نظر میاد پنی دوباره مغزشو ترکونده.»*

پنی: *«نه، نه. این بار داره کمک می‌کنه. یه جورایی.»*

شلدون سریع شروع به کشیدن یک فلوچارت روی وایت‌بردی که بالایش عبارت **«The Bazinga Categorizer»** نوشته شده است، می‌کند. راج زیر لب می‌گوید: *«خیلی داره لذت می‌بره.»* هاوارد هم چشم‌هایش را می‌چرخاند و شروع به غذا خوردن می‌کند.

چند دقیقه بعد، شلدون با غرور به سمت بقیه می‌چرخد.

شلدون: *«بفرمایید! یه سیستم بی‌نقص برای دسته‌بندی. بازینگا!»*

پنی با شک و تردید به تخته نگاه می‌کند.

پنی: *«صبر کن. واقعاً هواپیما رو گذاشتی توی دسته‌ی "اشیائی که به غرور شلدون تعلق ندارند"؟»*

لئونارد می‌خندد. هاوارد و پنی هم با هم دست می‌زنند. شلدون با اخم به همه نگاه می‌کند.

شلدون: *«هرچه‌قدر می‌خواید بخندید، ولی شما در حضور یه نابغه‌ی دسته‌بندی کلمات هستید!»*

پنی (با خنده): *«مرسی شلدون. به بچه‌ی دوستم می‌گم اسم این تکلیف رو ازت الهام بگیره، مثلاً *"دسته‌بندی بازینگا".»**

شلدون: *«بالاخره یه نفر به نبوغ من اعتراف کرد.»*

راج: «فکر کنم بهتره اسمشو بذاریم ** "The Big Bang Categorizer" **»

همه با سر تأیید می‌کنند و شلدون در حال غر زدن، به اتاقش می‌رود و زیر لب از قدرنشناسی بقیه شکایت می‌کند.

در این سوال قرار است شما تعدادی متن را از فایل‌هایی بخوانید و موضوعشان را حدس بزنید؛ به این صورت که در ابتدا موضوعات و کلمات مرتبط با هرکدام از آن‌ها را ورودی می‌گیرید. همراه هرکدام از این کلمات یک ضریب اعشاری داده می‌شود که به معنای میزان نزدیکی آن کلمه به آن موضوع است. **ارتباط یک موضوع با**

متن را برابر با جمع میزان ارتباط کلمه با آن موضوع، ضربدر تعداد تکرارهای آن کلمه در متن، برای تمامی کلمات مرتبط با آن موضوع تعریف می‌کنیم.

به عبارت دیگر اگر $R(T, w)$ میزان ارتباط کلمه‌ی w با موضوع T باشد و همچنین $f(w)$ تعداد تکرارهای w در متن و همچنین $S(T)$ مجموعه کلمات مرتبط با موضوع T باشد، داریم:

$$Relevance(T) = \sum_{w \in S(T)} f(w) \times R(T, w)$$

با توجه به رابطه‌ی بالا، موضوع با بیشترین نزدیکی به متن را خروجی دهید.

ورودی

هنگام اجرای برنامه، آدرس نسبی فایل Category (که قالب محتوای آن در ادامه آمده است) و آدرس فایل‌های متنی‌ای که قرار است موضوع آن‌ها را تعیین کنید، توسط آرگومان‌های خط فرمان (`argv` , `argc`) داده می‌شوند. (دقت کنید که ابتدا آدرس فایل Category و سپس آدرس دیگر فایل‌ها، یکی یکی داده می‌شوند.)

▼ توضیحاتی درباره‌ی argv و argc

هنگام اجرای برنامه‌ی کامپایل شده، می‌توانیم عبارت‌هایی را به برنامه ورودی دهیم که از طریق آرگومان‌های argv و argc درون برنامه در دسترس خواهند بود.

۱. **`argc`** :
   - یک عدد صحیح است که نشان‌دهنده‌ی تعداد آرگومان‌های خط فرمان است.
   - این مقدار شامل نام خود برنامه نیز می‌شود. (همواره بیشتر یا مساوی یک است)

۲. **`argv`** :
   - یک آرایه از رشته‌ها (آدرس‌های متنی) است که هر رشته یکی از آرگومان‌های خط فرمان را نگه می‌دارد.
   - اولین مقدار `argv` ، همیشه نام برنامه (مسیر فایل اجرایی) است.

برای مثال اگر در ترمینال برنامه‌ای را به صورت زیر اجرا کنیم، خواهیم داشت:

```
./example categories.txt file1.txt file2.txt
```

```
argc = 3
argv[0] = "./example"
argv[1] = "categories.txt"
argv[2] = "file1.txt"
argv[3] = "file2.txt"
```

محتوای داخل فایل Category در قالب زیر است:

```
category1:keyword1:coefficient1:keyword2:coefficient2...
category2:keyword1:coefficient1:keyword2:coefficient2...
.
.
.
```

در هر خط، یک موضوع و کلیدواژه‌های آن به‌همراه ضریب مربوط به آن‌ها در قالب بالا وجود دارد.

تضمین می‌شود که همه‌ی واژه‌های فایل Category با حروف کوچک باشند و کاراکتر فاصله در کلیدواژه‌ها وجود نداشته باشد.

$$NumberofCategories \leq 10$$

$$KeywordsinEachCategory \leq 40$$

$$LineLength \leq 1024$$

توجه: دقت کنید که در پردازش فایل‌های متنی که باید موضوع آن‌ها را تعیین کنید، واژه‌ها علاوه بر کارکتر فاصله، می‌توانند توسط هر یک از علائم نگارشی `\t\n.,;!?\"'()[]{}:-` جدا شده باشند.

خروجی

برای یافتن موضوع هر متن، کافی است وزن واژه‌های کلیدی که در آن متن ظاهر شده‌اند (که می‌توانند با حروف بزرگ یا کوچک باشند) را برای موضوعات مختلف حساب کنید و موضوعی که طبق رابطه‌ی گفته شده در بالا بیشترین وزن را دارد، به عنوان موضوع متن خروجی دهید. اگر برای یک متن، چند موضوع وزن‌های

یکسان داشتند و از بقیه بیشتر بودند، عبارت `Unknown` را چاپ کنید. شما باید در هر خط خروجی، به ترتیب موضوع متن فایل‌هایی که داده شده است را چاپ کنید.

تست و ارسال

شما می‌توانید نمونه‌ی شامل چند تست‌کیس و داوری لوکال را از این‌جا دریافت کنید.

داوری لوکال صرفاً برای آزمایش خودتان است و در نهایت باید فایل main.c خود را در کوئرا بارگذاری کنید و نمره در کوئرا ملاک ارزیابی خواهد بود.

File System

- محدودیت زمان: ۵ ثانیه

در شبی آرام، شلدون کوپر در آپارتمانش نشسته و با دقت به صفحه‌ی نمایش لپ‌تاپ خیره شده بود. دوستانش، لئونارد، راج و هاوارد، او را تماشا می‌کردند. شلدون ناگهان از جایش بلند شد و گفت: «من به نتیجه رسیدم. ما به کمک نیاز داریم! و نه فقط از هر کسی، بلکه از ذهن‌های جوان و درخشان دانشجویان. این یک مسئله‌ی حیاتی برای پیشرفت بشر است.»

لئونارد کنجکاو پرسید: «کمک برای چه چیزی؟» شلدون با جدیت پاسخ داد: «برای ساخت سامانه‌ای که بتواند ساختار دنیای دیجیتال را بهتر از همیشه مدیریت کند. اما این کاری نیست که بتوانیم به تنهایی انجام دهیم. ما به دانشجویان نیاز داریم، به افرادی با خلاقیت و اشتیاق برای حل مسائل علمی.»

او رو به دوستانش کرد و ادامه داد: «فقط تصور کنید، جوانانی که در این پروژه شرکت می‌کنند، ممکن است روزی تاریخ‌ساز شوند. این یک فرصت نادر برای همکاری با ذهن‌های بزرگ‌تر است.»

شلدون سپس رو به جمعی از دانشجویان فرضی کرد و گفت: «دانشجویان عزیز، ما از شما کمک می‌خواهیم. شما قرار است بخشی از پروژه‌ای باشید که می‌تواند شیوه‌ی مدیریت داده‌ها را دگرگون کند. این فرصتی است تا با ایده‌ها و مهارت‌های خودتان، بخشی از یک پروژه‌ی علمی بزرگ‌تر شوید. شاید این پروژه ساده به نظر

برسد، اما اهمیت آن در تأثیرش بر آینده پنهان است. شما می‌توانید بخشی از این حرکت باشید، جایی که دانش و خلاقیت به هم می‌پیوندند.»

هاوارد به شوخی اضافه کرد: «و البته، کار با شلدون همیشه داستان‌های خنده‌دار زیادی برای تعریف کردن باقی می‌گذارد.» شلدون با نگاه تیزش به هاوارد ادامه داد: «این کار به چیزی فراتر از شوخی نیاز دارد. این فرصتی است برای شما که توانایی‌هایتان را به نمایش بگذارید و بخشی از تغییری باشید که دنیا به آن نیاز دارد.»

راج گفت: «پس اگر آماده هستید، ما اینجا هستیم تا شما را راهنمایی کنیم. بیایید باهم آینده را بسازیم.»

در این سوال شما باید برخی از عملیات ساده‌ی مربوط به مدیریت فایل‌ها و پوشه‌ها (در سیستم `POSIX`) را در زبان سی پیاده‌سازی کنید. پیشنهاد می‌شود قبل از شروع به پیاده‌سازی این تمرین، در مورد توابع موجود در کتابخانه‌های `unistd.h` ، `dirent.h` و `sys/stat.h` تحقیق کنید. همچنین در مورد آدرس‌دهی نسبی (Relative)، آدرس‌دهی مطلق (Absolute) و این‌که چگونه می‌توان از این دو حالت برای آدرس‌دهی فایل‌ها استفاده کرد نیز مطالعه کنید.

قبل از پیاده‌سازی، هشدار انتهای متن سوال را بخوانید!

استفاده از تابعِ system یا موارد مشابه که از دستورات shell استفاده می‌کنند مجاز نیست و در صورت هرگونه استفاده، نمره‌ی کل سوال را از دست خواهید داد.

اجرای برنامه

همانند باقی سوالات تمرین، در این سوال نیز باید از argc و argv استفاده کنید؛ به‌طوری‌که یک آرگومان به فایل اجرایی برنامه‌ی شما داده می‌شود و آن هم آدرس یک فایل متنی حاوی دستوراتی است که کد شما باید به ترتیب آن دستورات را اجرا کرده و تغییرات لازم در فایل‌ها و دایرکتوری‌ها را بوجود آورد یا در صورت لزوم، خروجی‌های احتمالی را در stdout چاپ کند.

برای مثال، برنامه‌ی شما می‌تواند با دستور زیر اجرا شود:

```
./main.out start.txt
```

در این حالت برنامه‌ی شما باید فایل start.txt را باز کند و دستورات موجود در آن را (که در ادامه توضیح داده خواهد شد) به ترتیب اجرا کند.

نکته: تضمین می‌شود که موقعیت Shell هنگام شروع اجرای برنامه، در محل فایل اجرایی قرار دارد. این مکان را از این پس base folder می‌نامیم. همچنین تضمین می‌شود تمام دستورات در base folder و فرزندانش داده خواهند شد.

دستورات

در فایلی که به‌عنوان **فایل دستورات** به برنامه‌ی شما داده می‌شود، در هر خط یک دستور قرار گرفته است که لیست و جزئیات دستورات در ادامه آورده شده است.

`whereami`

این دستور باید موقعیت نسبی (Relative Address) فعلی‌تان نسبت به پوشه‌ای که فایل اجرایی برنامه درون آن قرار دارد (base folder) را در خروجی چاپ کند. منظورمان از موقعیت فعلی، آدرس نسبی **فایل دستوراتی** که هم‌اکنون در حال اجرای دستورات آن هستیم است.

```
whereami
```

مثال: اگر فایل دستوراتی که در حال اجرای دستورات درون آن هستیم، در آدرس نسبی

```
./folder/sub/setup.cmd
```

قرار داشته باشد، خروجی دستور به صورت زیر خواهد بود: (خروجی باید بدون `./` در ابتدا و `/` در انتها چاپ شود)

```
folder/sub
```

و تنها در حالتی که در محل اولیه قرار داشته باشیم (base folder)، خروجی این دستور یک نقطه `.` خواهد بود.

`source`

قالب دستور:

```
source ../some_folder/some_command.cmd
```

کارکرد دستور: این دستور، یک **فایل دستورات** دیگر که آدرس نسبی آن داده می‌شود، (که ممکن است حاوی `..` و `.` باشد و نسبت به موقعیتِ فایلِ دستوراتِ فعلی داده می‌شود) را به‌صورت بازگشتی اجرا می‌کند. یعنی مانند آغاز، دستورات موجود در **فایل دستورات** داده شده را خط به خط اجرا می‌کند و پس از تمام شدن اجرای آن، به ادامه‌ی دستورات فایل فعلی برمی‌گردد.

راهنمایی: `..` به معنای دایرکتوری والد و `.` به معنای دایرکتوری جاری است.

`create-dir`

قالب دستور:

```
create-dir <dir_name>
```

کارکرد دستور: با این دستور، دایرکتوری با نام مشخص شده در مکان فعلی ساخته می‌شود.

تضمین می‌شود که نام پوشه حاوی کاراکترهای خاص و اسپیس نیست و همچنین دستور ساخت بازگشتی پوشه داده نمی‌شود؛ برای مثال دستور `create-dir f1/f2` داده نخواهد شد.

خطا: اگر فایل یا دایرکتوری‌ای با نام گفته‌شده در مکان فعلی از قبل وجود داشت، خطای زیر باید در خروجی چاپ شود:

```
A file or directory exists with name <dir_name>
```

`create`

قالب دستور:

```
create <path> <file_name>
<file-content-line-1>
<file-content-line-2>
....
!end!
```

کارکرد دستور: با این دستور، یک فایل با نام داده شده در مسیر نسبی `path` ساخته می‌شود. (برای مثال اگر `path` یک نقطه (`.`) باشد، فایل در **مسیر فعلی** ساخته خواهد شد). نام فایل ممکن است شامل پسوند باشد. محتویاتی که در خطوط بعدی دستور خواهند آمد به‌صورت متنی خط به خط درون آن فایل نوشته خواهند شد و با دیدن `!end!` فایل بسته می‌شود. (این خط نباید در فایل ذخیره شود)

تضمین می‌شود `path` داده شده، آدرسی موجود و معتبر است و نیاز به ساخت آن نباشد.

خطا: اگر فایل یا دایرکتوری‌ای با نام گفته‌شده در مکان فعلی از قبل وجود داشت، خطای زیر باید در خروجی چاپ شود:

```
A file or directory exists with name <file_name>
```

`delete`

قالب دستور:

```
delete <name>
```

کارکرد دستور: این دستور، فایل/دایرکتوری گفته شده که در مکان فعلی وجود دارد را پاک می‌کند. اگر name مربوط به یک دایرکتوری باشد، دایرکتوری را به همراه همه‌ی فایل‌های داخل آن پاک می‌کند.

راهنمایی: شما باید این دستور را به‌صورت بازگشتی پیاده‌سازی کنید.

خطا: اگر فایل/دایرکتوری با نام گفته شده در مکان فعلی وجود نداشت، خطای زیر باید در خروجی چاپ شود:

```
File or directory with name <name> doesn't exists!
```

`copy`

قالب دستور:

```
copy <src-path> <dest-path>
```

کارکرد دستور: این دستور، فایل/دایرکتوری با مسیر داده‌شده به‌عنوان `src-path` را در داخل مسیر داده‌شده به‌عنوان `dest-path` کپی می‌کند.

نکته: آدرس‌ها به‌صورت **نسبی از موقعیت فعلی** داده می‌شوند.

نکته: تضمین می‌شود آدرس `dest-path` آدرس نسبی صحیح یک دایرکتوری است و وجود دارد؛ و همچنین تضمین می‌شود در صورتی‌که `src-path` دایرکتوری باشد، `dest-path` از زیرمجموعه‌های آن نیست.

راهنمایی: در صورتی که `src-path` یک دایرکتوری باشد، شما باید یک دایرکتوری با همان نام در `dest-path` بسازید و به‌صورت بازگشتی محتوای آن را کپی نمایید.

خطا: اگر فایل/دایرکتوری معرفی شده به‌عنوان `src-path` وجود نداشت، خطای زیر باید در خروجی چاپ شود:

```
Invalid source path!
```

اگر در دایرکتوری مقصد فایل/دایرکتوری با نام مشابه مبدأ وجود داشت، خطای زیر باید در خروجی چاپ شود:

```
A file with this name already exists in the destination!
```

`move`

قالب دستور:

```
move <src-path> <dest-path>
```

کارکرد دستور: این دستور، مسیر فایل/دایرکتوری به‌عنوان `src-path` را به `dest-path` تغییر می‌دهد. یعنی همزمان move و rename را انجام می‌دهد. (این به این معنی است که در `dest-path` مسیر کامل داده می‌شود که شامل نام جدید نیز می‌شود.)

نکته: آدرس‌ها به‌صورت نسبی از **موقعیت فعلی** داده می‌شوند.

نکته: تضمین می‌شود دایرکتوری والدِ `dest-path` (یعنی با حذف نام جدید از آن)، آدرس نسبی صحیح یک دایرکتوری است و وجود دارد. همچنین تضمین می‌شود در صورتی‌که `src-path` دایرکتوری باشد، `dest-path` از زیرمجموعه‌های آن نیست.

خطا: اگر فایل/دایرکتوری معرفی شده به‌عنوان `src-path` وجود نداشت، خطای زیر باید در خروجی چاپ شود:

```
Invalid source path!
```

اگر فایل/دایرکتوری‌ای با مسیر `dest-path` از قبل وجود داشت، خطای زیر باید در خروجی چاپ شود:

```
A file with this name already exists in the destination!
```

`tree`

قالب دستور:

```
tree
```

کارکرد دستور: این دستور، درخت فایل سیستم موجود در **موقعیت فعلی** را در خروجی چاپ می‌کند. دقت کنید که ترتیب نمایش Entryها مهم است؛ باید ابتدا در هر لایه Entry ها به ترتیب الفبا (Lexicographic) مرتب شوند.

کاراکترهای موردنیاز: شما باید در پیاده‌سازی این درخت، از علائم زیر استفاده کنید:

- Final Entry: `└──`
- Middle Entry: `├──`
- No Entry (Connector): `│`

پس از هر connector از یک کاراکتر `\t` و میان کاراکتر Entry و نام آن از یک space برای ایجاد فاصله‌ی مناسب استفاده کنید.

راهنمایی: در پیاده‌سازی درخت به‌صورت بازگشتی، حواستان به Final بودن یا نبودن Entryهای لایه‌های قبلی باشد. دقت کنید پس از آمدن Final Entry در یک سطح، در خطوط بعدی در آن سطح connector نداریم و در صورت نیاز یک کاراکتر space جایگزین آن می‌شود.

همچنین می‌توانید از کدهای موجود در مخزن۱ و مخزن۲ ایده بگیرید.

خروجی نمونه:

```
├── afile0_1
├── bdirectory0_1
│    ├── afile1_1
│    ├── bdirectory1_2
│    │    └── file2_1
│    ├── cdirectory1_3
│    │    └── directory2_1
│    │         └── file3_1
│    ├── dfile1_2
│    └── efile1_3
└── cfile0_2
```

هشدار

دقت کنید که در هنگام کار بر روی کد این سوال، در حال تعامل با فایل‌سیستم اصلی خود هستید؛ بنابراین هنگام زدن کدها دقت کنید که به فایل‌سیستم خود آسیب نزنید. یکی از نکاتی که باید خیلی به آن توجه کنید، این است که هیچ یک از آدرس‌هایی که ما در این سوال با آن‌ها کار می‌کنیم آدرس‌های مطلق نیستند (یعنی با `/` شروع نمی‌شوند)؛ بنابر این اگر قسمتی از پیاده‌سازی سؤال، نیاز بود با آدرس مطلق کار کنید، باید حتماً با توابع string آن را به آدرس نسبی مناسب تبدیل کنید.

تست و ارسال

شما می‌توانید نمونه‌ی شامل چند تست‌کیس و داوری لوکال را از این‌جا دریافت کنید.

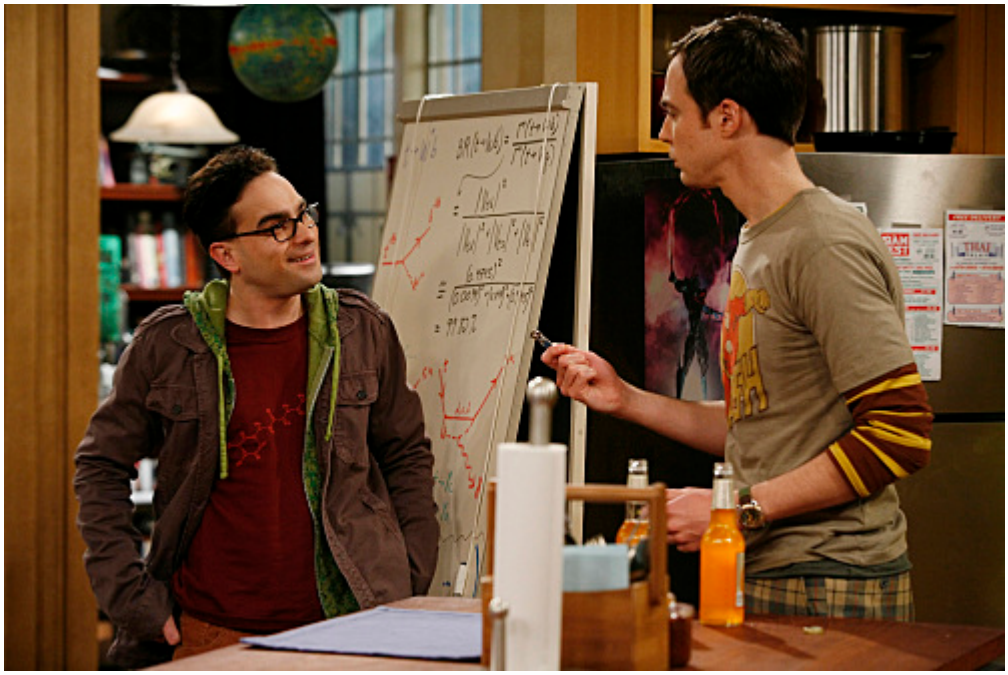
داوری لوکال صرفاً برای آزمایش خودتان است و در نهایت باید فایل main.c خود را در کوئرا بارگذاری کنید و نمره در کوئرا ملاک ارزیابی خواهد بود.

WAV-ing در فیزیک

- محدودیت زمان: ۵ ثانیه

شلدون پشت میز نشسته بود و با دقت به کامپیوترش نگاه می‌کرد. با صدایی پر از هیجان گفت:

«آقایان! امروز وارد دنیای شگفت‌انگیز امواج صوتی می‌شویم. جهانی به‌اندازه‌ی مکانیک کوانتوم جذاب!»



لئونارد آهی کشید و گفت:

«شلدون، لطفاً بگو که این یکی از اون آزمایش‌های پیچیده‌ی همیشگیت نیست.»

شلدون با غرور لبخندی زد و گفت:

«اصلاً و ابداً. این یه Exploration منطقی درباره‌ی امواج صوتی ضبط‌شده به‌صورت فایل‌های WAV است که می‌تونن اسرار جهان رو فاش کنن.»

پنی سرش را داخل آورد و با یک لیوان اسموتی در دستش گفت:

«بذار حدس بزنم شلدون، داری سعی می‌کنی صدات رو به یه جور سلاح تبدیل کنی؟»

شلدون پوزخندی زد و گفت:

«سلاح؟ پنی، مسخره نباش. من فقط دارم فایل‌های صوتی رو بررسی می‌کنم تا رزونانس خاص تارهای صوتی انسان رو بفهمم. این علمه!»

هاوارد با خنده گفت:

«بذار حدس بزنم، اول با صدای خودت شروع کردی. قراره کنسرت "بهترین‌های شلدون" رو بشنویم؟»

شلدون به حرفش اعتنا نکرد و دکمه‌ی پخش را روی کامپیوتر فشار داد. صدای بلند *«بازینگا!»* ازبلندگوها پخش شد و آپارتمان را لرزاند.

راج گوش‌هایش را گرفت و گفت:

«شلدون! این از بوق ماشین هاوارد هم بلندتر بود!»

شلدون با افتخار گفت:

«دقیقاً! دامنه و فرکانس صدای من به طور دقیق برای حداکثر اثرگذاری تنظیم شده بود. حالا بیاید ببینیم می‌تونید الگوهای پنهان رو از این فایل WAV استخراج کنید.»

لئونارد با چشمانی نیمه‌باز گفت:

«و دقیقاً چرا باید این کار رو انجام بدیم؟»

شلدون به سمتش خم شد و با صدای آرام و مارموز گفت:

«چون لئونارد، ممکنه رازهای جهان در ارتعاشات صدا نهفته باشه. کی نمی‌خواد تو دل واقعیت WAV-ing کنه؟»

در این تمرین بسیار جذاب، شما قرار است بر روی نوع خاصی از فایل‌های صوتی (فایل‌های صوتی با فرمت `.wav`) کار کنید و یک ویرایشگر صوتی ساده بسازید. برنامه‌ی شما باید دستورات مختلفی را که به آن به صورت آرگومان داده می‌شود، روی فایل صوتی اجرا کند.

ساختار فایل WAV

ابتدا بیایید با ساختار فایل‌های صوتی با فرمت `.wav` اندکی آشنا شویم.

فرمت `.wav` در فایل‌های صوتی، یکی از فرمت‌های رایج برای ذخیره‌سازی اطلاعات صوتی است که از استاندارد (Resource Interchange File Format) **RIFF** برای ساختار داخلی خود استفاده می‌کند. فایل‌ها با این فرمت معمولاً شامل بخش‌هایی به نام "chunk" هستند که هر کدام اطلاعات خاصی را ذخیره می‌کنند.

به طور کلی فرمت فایل WAV می‌تواند به‌صورت **فشرده** و **غیرفشرده (PCM)** باشد. کار با داده‌های فایل‌هایی که حاوی فشرده‌سازی هستند به‌صورت مستقیم و بدون کتابخانه، کار چندان راحتی نیست؛ ولی

در فرمت PCM (که نوع رایج فایل wav است)، داده‌ها به‌صورت خام ذخیره می‌شوند و به‌صورت مستقیم قابل پردازش هستند.

در این تمرین، ما تمرکز خود را بر روی مدل **غیر فشرده (PCM)** قرار می‌دهیم که داده‌های صوتی در آن به صورت دنباله‌ای از نمونه‌ها (samples) با دقت‌های مختلف ذخیره می‌شوند. این نوع ذخیره‌سازی باعث می‌شود که کیفیت صدا حفظ شود، اما حجم فایل نسبتاً بالا باشد.

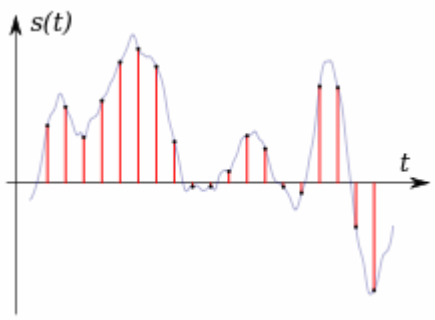
توضیح مدل PCM و ارتباط آن با صدا

در مدل (Pulse Code Modulation) **PCM**، صدا از طریق تبدیل سیگنال آنالوگ به دیجیتال، به‌صورت دنباله‌ای از **نمونه (sample)های دیجیتال** ذخیره می‌شود. هر نمونه نشان‌دهنده‌ی یک **لحظه‌ی خاص** از سیگنال صوتی است که مقدار **شدت فشار هوا یا دامنه (Amplitude)** در آن لحظه را اندازه‌گیری می‌کند.

تصور کنید شما در حال ضبط صدای یک نوت موسیقی هستید. این صدا به صورت یک **موج صوتی** از تغییرات فشار هوا (امواج صوتی) در فضا ایجاد می‌شود. در PCM، این موج به یک سری عدد تبدیل می‌شود که هر عدد نشان‌دهنده‌ی **جابه‌جایی یا فشردگی هوا** در یک لحظه‌ی زمانی خاص است. برای مثال، در هر لحظه یک **نمونه (sample)** به‌عنوان نماینده‌ی شدت فشار هوا (یا همان دامنه‌ی صوتی) ثبت می‌شود.

- **مفهوم عمق بیت (bit depth):** این عدد نشان‌دهنده‌ی تعداد بیت هر نمونه (sample) است؛ اگر عمق بیت ۱۶ باشد، یعنی هر نمونه با یک عدد ۱۶ بیتی ثبت می‌شود.
- **فرکانس نمونه‌برداری (Sampling Rate):** بیانگر تعداد دفعات نمونه‌گیری از موج در هر ثانیه است؛ برای مثال، فرکانس نمونه‌برداری ۴۴۱۰۰ هرتز، به این معنی است که در هر ثانیه ۴۴۱۰۰ نمونه از موج صوتی گرفته می‌شود. تعدد نمونه‌ها برای ثبت و بازسازی دقیق موج صوتی ضروری است.

به عبارت ساده‌تر، هر **نمونه** در PCM همانند یک عکس از موج صوتی در یک لحظه‌ی زمانی است. هرچه **فرکانس نمونه‌برداری** بالاتر باشد، دقیق‌تر می‌توانیم موج صوتی را ثبت کنیم. اگر فرکانس نمونه‌برداری پایین باشد، ممکن است جزئیات مهمی از صدا گم شود و کیفیت صدای ضبط‌شده کاهش یابد.



ارتباط نمونه‌ها با فرکانس صدا

اگر فرض کنیم شما در حال ضبط صدای یک نوت موسیقی با فرکانس ۴۴۰ هرتز (نت "لا") هستید، این یعنی در هر ثانیه ۴۴۰ بار نوسان موج صوتی را شاهد خواهیم بود. برای این‌که بتوانیم این نوسان سینوسی را با تقریب خوبی به‌صورت دقیق ثبت کنیم، لازم است فرکانس نمونه‌برداری بسیار بیشتر از این عدد باشد تا بتواند نمونه‌های لازم که نشان‌دهنده‌ی وضعیت‌های هر لحظه از این نوسان است را ثبت کند (واضح است که برای ثبت حداقلیِ این نوسان، باید فرکانس نمونه‌برداری حداقل **دو برابر** این مقدار، یعنی ۸۸۰ هرتز باشد). البته معمولاً برای دقت بیشتر و حفظ کیفیت صدا، از فرکانس‌های بالاتر مثل ۴۴۱۰۰ هرتز استفاده می‌شود.

حال اگر بخواهیم از روی تعدادی از نمونه‌ها به فرکانس آن لحظات پی‌ببریم، باید از **تبدیل فوریه‌ی گسسته (Discrete Fourier Transform)** استفاده کنیم. البته در این تمرین شما درگیر این مسئله نمی‌شوید و کد تابع مربوط به **تبدیل تعدادی نمونه به فرکانس** در قالبی که در انتهای صورت سوال در اختیار شما قرار گرفته، پیاده‌سازی شده است و شما صرفاً باید آن را به‌درستی فراخوانی کنید.

سرلوحه (Header) فایلِ WAV

اگر به‌صورت باینری و بایت‌به‌بایت به محتوای یک فایل `.wav` نگاه کنیم، همیشه می‌بینیم بخشی از ابتدای فایل‌ها از ساختار مشابهی برخوردارند. این ساختار را در جدول زیر مشاهده می‌کنید. (همچنین می‌توانید از این لینک نیز این ساختار و توضیحات مربوط به آن را ببینید).

هدر فایل WAV (۴۴ بایت اولیه) شامل اطلاعات زیر است:

فیلد	شماره بایت	توضیحات
Chunk ID	3–0	مقدار ثابت "RIFF"
Chunk Size	7–4	تعداد بایت‌های Chunk پیش‌رو (اندازهٔ فایل منهای ۸)
Format	11–8	مقدار ثابت "WAVE"
Subchunk1 ID	15–12	مقدار ثابت "fmt "
Subchunk1 Size	19–16	اندازه بخش فرمت (معمولاً 16 برای PCM)

فیلد	شماره بایت	توضیحات
Audio Format	21–20	1 برای PCM (غیر فشرده)
Num Channels	23–22	تعداد کانال‌ها (1 = مونو، 2 = استریو)
Sample Rate	27–24	نرخ نمونه‌برداری (به‌طور مثال، 44100 هرتز)
Byte Rate	31–28	تعداد بایت پردازش شده در ثانیه ($(SampleRate * BlockSize$)
Block Size	33–32	تعداد بایت‌های هر نمونه ($(NumChannels * BitsPerSample/8$)
Bits per Sample	35–34	تعداد بیت‌های هر نمونه صوتی (۸، ۱۶، ۲۴ یا ۳۲ بیت)
Subchunk2 ID	39–36	مقدار ثابت "data"
Subchunk2 Size	43–40	تعداد بایت داده‌های خام صوتی (که در ادامه می‌آید)
ادامه	Wave data	توضیحات در ادامه

بخش داده

- این قسمت شامل داده‌های خام صوتی است.
- داده‌ها بسته به فرمت ممکن است ۸، ۱۶، ۲۴ یا ۳۲ بیتی باشند؛ و همچنین تعداد $Channel$ ها ممکن است متفاوت باشد (mono و stereo)؛ بنابراین به‌طورخلاصه: اگر به مقدار $BlockSize$ از فایل بخوانید.، یک نمونه (Sample) را خوانده‌اید. تعیین این‌که چقدر باید خوانده شود نیز با توجه به جدول فوق، می‌تواند با استفاده از Chunk size یا Subchunk2 size تعیین شود.

توجه: مواردی که چند بایتی هستند را باید به صورت **Little Endian** لحاظ کنید.

توجه: اگر اندکی به جدول بالا دقت کنید، در می‌یابید که مواردی هستند که با روابطی به یکدیگر مربوط می‌شوند. اگر چنین رابطه‌ای در یک فایل `.wav` برقرار نباشد، می‌گوییم فایل Corrupt شده است.

تضمین می‌شود که به ورودی برنامهٔ شما فایل سالم و غیر Corrupt داده می‌شود. ولی شما باید دقت کنید در فایل‌هایی که خروجی می‌دهید آن موارد را رعایت کنید.

این وب‌سایت نیز توضیحاتی کامل و روان دربارهٔ فرمت فایل‌های WAV دارد که در صورت لزوم می‌توانید به آن مراجعه کنید.

دستورات برنامه

به‌صورت کلی، برنامهٔ شما چنین چیزی کارکردی خواهد داشت:

```
Usages: ./wav_proc <input_wave_file> <command> [output_file]
Commands:
    --info              Display information about the wave file
    --reverse           Reverse the wave file
    --gender            Determine the gender of the speaker in the wave fil
    --double-speed      Double the speed of the wave file
    --double-speed-safe Double the speed (lossless)
```

(به‌جای `./wav_proc` نام فایل اجرایی برنامهٔ شما - به‌صورت دقیق‌تر `argv[0]` - خواهد بود؛ در ادامهٔ صورت سوال هرجا از این اسم استفاده شده منظور همان فایل اجرایی برنامه شما می‌باشد)

ویرایشگر صوتی ساده شما باید از چند دستور که از طریق آرگومان‌های خط فرمان (argv) داده می‌شود و توضیحات آن‌ها در ادامه می‌آید، پشتیبانی کند.

توجه کنید که لازم نیست کد برنامه را از صفر پیاده‌سازی کنید؛ شما می‌توانید تمپلیت کد این‌تمرین به همراه تست‌کیس‌ها و نیز اسکریپت جاج لوکال را از انتهای این صفحه دانلود کرده و صرفاً باید بخش‌هایی که با TODO علامت‌گذاری شده‌اند را تکمیل کنید.

دستور `info`

با زدن دستور زیر در ترمینال، باید اطلاعات فایل صوتی نمایش داده شود:

```
./wav_proc <wav_file_path> --info
```

خروجی این دستور در ترمینال (`stdout`) چاپ می‌شود و باید طبق توضیحات زیر باشد و دو حالت دارد:

حالت اول: اگر `Audio Format` برابر 1 بود، یعنی فایل صوتی از نوع PCM است؛ و باید اطلاعات را به‌فرمت زیر چاپ کنید:

```
Audio Format: PCM
Channels: <channels>
<samplerate>Hz - <bits> bit
Length: <length> seconds
```

به فاصله‌ها و حروف دقت کنید و صرفاً جای `<channels>` و `<samplerate>` و `bits` و `length` موارد زیر را قرار دهید:

- **`<chennels>`** : اگر تعداد کانال‌ها برابر یک باشد، `Mono` ، اگر برابر دو بود `Stereo` و در غیر این‌صورت عدد آن را چاپ کنید.
- **`<samplerate>`** : عدد مربوط به `Sample Rate` (مثال: 32000, 8000, 44100 و ...)
- **`<bits>`** : عمق بیت هر نمونه (مثال: 16, 32, 8 و ...)
- **`<length>`** : مدت زمان فایل صوتی به واحد ثانیه. این عدد را باید از روی اطلاعاتی که در اختیار دارید محاسبه کنید.

حالت دوم: اگر `Audio Format` برابر 1 نبود، بر اساس مقادیر زیر صرفاً تک خط زیر را خروجی دهید:

```
Audio Format: <format>
```

مقدار	format
3	`IEEE 754 Float`
6	`A-Law`
7	`Mu-Law`
هرمقدار دیگر	`Unknown`

دستور `reverse`

با زدن این دستور بسیار جذاب در ترمینال، فایل صوتی شما برعکس می‌شود!

```
./wav_proc <wav_file_path> --reverse <output>
```

- دقت کنید که خروجی را باید داخل فایل با مسیر `<output>` ذخیره کنید و فایل ورودی را نباید تغییر دهید.
- برای معکوس کردن صوت، باید داده‌های بخش `data` از انتها به ابتدا نوشته شوند؛ اما باید حواستان به Block-by-Block بودن نمونه‌ها باشد؛ چرا که محتویات درونی یک Block نباید معکوس شود در غیراین‌صورت فایل شما خراب خواهد شد! برای مثال، برای فایل‌های استریو، بلوک‌های هر کانال باید به صورت هماهنگ معکوس شوند. (یادآوری: تعداد بایت هر بلاک داخل `Block Size` در هدر مشخص شده است)

دستور `gender`

این دستور جادویی باید تعیین کند کسی که صدای او در فایل صوتی است **زن** یا **مرد** است!

```
./wav_proc <wav_file_path> --gender
```

خروجی دستور باید به فرمت زیر در ترمینال چاپ شود:

```
<Gender> (Value: <value>)
```

که `<value>` یک عدد اعشاری است که باید با **دو رقم اعشار** چاپ شود و نیز `<Gender>` یا `Male` یا `Female` چاپ خواهد شد که روش بدست آوردن مقدار و جنسیت در ادامه آمده است:

توضیحات:

برای این‌که امکان تشخیص جنسیت صدا میسر باشد، باید داده‌های خام صوتی را (که در حوزه زمان هستند) به حوزهٔ فرکانس ببریم. (یعنی همان‌طور که در این‌جا توضیح داده شد، ما تعدادی Sample در اختیار داریم که برای این‌که بتوانیم به فرکانس دست پیدا کنیم باید از تبدیل فوریهٔ گسسته استفاده کنیم).

شما درگیر جزئیات پیاده‌سازی بخش تبدیل فوریه نخواهید شد و در تمپلیتی که در اختیارتان قرار گرفته است، تابع `fft` به‌طور کامل و تابع `apply_fourier` نیز به‌صورت تقریباً کامل پیاده‌سازی شده است. (صرفاً سه خط ابتدایی تابع `apply_fourier` را باید کامل کنید که مربوط به جزئیات فرمت wav هستند)

ولی شما باید دقیق بدانید که چه چیزهایی را باید به تابع `apply_fourier` ورودی بدهید و چه چیزی به شما خروجی خواهد داد. یعنی به‌طور خلاصه باید بدانید چگونه از این تابع در راستای بدست آوردن جنسیت صدا استفاده کنید.

```
unsigned int apply_fourier(FourierElement* dest, void *wave_data, unsi
    unsigned int num_channels, unsigned int bit_depth, unsigned int sa
```

تابع `apply_fourier` به تعداد ۶ ورودی دارد:

- **آرگومان `dest` که از نوع `FourierElement*` است**: آرایهٔ مقصد که باید به تعداد کافی از قبل allocate شده باشد.
- **آرگومان `wave_data` که از نوع `void*` است**: داده‌های خام موجود در بخش Data در فایل صوتی.
- **آرگومان `size`** : تعداد بایت داده در `wave_data`
- **آرگومان‌های `num_channels` ، `bit_depth` و `sample_rate`** : با توجه به هدر فایل wav باید تنظیم شوند.

مقدار return value تابع نیز تعدادِ `FourierElement` های نوشته شده در آرایهٔ `dest` است.

هر `FourierElement` شامل `frequency` (عدد فرکانس) و `magnitude` (شدت فرکانس) است. شما پس از فراخوانی صحیح تابع `apply_fourier` ، تعداد زیادی زوج مرتب (f, m) در اختیار خواهید داشت که در آرایهٔ `dest` شما قرار دارند. (تذکر مجدد: آرایهٔ مقصد باید قبل از فراخوانی تابع، به‌میزان کافی Allocate شده باشد. میزان کافی برابر است با تعداد Sample Block های موجود در فایل صوتی.)

حال باید مراحل زیر را پیاده‌سازی کنید:

۱. به‌ازای همهٔ (f, m) هایی که $f \in [85, 190]$ ، میانگین m را به‌دست آورید. (این عدد را `male_value` می‌نامیم)

۲. به‌ازای همهٔ (f, m) هایی که $f \in [165, 300]$ ، میانگین m را به‌دست آورید. (این عدد را `female_value` می‌نامیم)

۳. در صورتی که `male_value` از `female_value` بیشتر بود، Gender برابر `Male` و در غیر این‌صورت برابر `Female` می‌باشد.

۴. برای بدست آوردن `value` کافی‌است از بین این دو مقدار، عدد بزرگ‌تر را بر عدد کوچکتر تقسیم کرده و با دو رقم اعشار طبق فرمتی که بالاتر گفته شد چاپ کنید.

دستور `double-speed`

این دستور باید سرعت فایل صوتی ورودی را دو برابر کرده و در فایل خروجی ذخیره کند!

دو حالت/روش برای این دستور وجود دارد که در ادامه توضیحات آن‌ها آورده شده است.

```
./wav_proc <wav_file_path> --double-speed <output>
./wav_proc <wav_file_path> --double-speed-safe <output>
```

- دقت کنید که خروجی را باید داخل فایل با مسیر `<output>` ذخیره کنید و فایل ورودی را نباید تغییر دهید.

توضیحات:

برای ایجاد تغییرات این‌چنینی روی یک فایل صوتی، چند حالت وجود دارد؛

- تغییر Tempo (صرفاً تغییر سرعت به‌گونه‌ای که تنِ صدا تغییر نکند)
- تغییر Pitch (صرفاً تغییر تن صدا به‌گونه‌ای که سرعت پخش تغییر نکند)
- تغییر Speed (افزایش/کاهش سرعت همراه با افزایش/کاهش تن صدا)

با توجه به‌اینکه پیاده‌سازی دو مورد اول با چالش‌ها و جزئیاتی همراه است؛ بنابراین ما در هر دو روش عادی و safe این دستور صرفاً مورد سوم را مد نظر داریم.

اکنون دو نوعِ دستور را با هم بررسی می‌کنیم:

- نوع اول، عادی

با توجه به مطالبی که در این‌جا خواندیم، در می‌یابیم که یکی از راه‌هایی که از طریق آن می‌توانیم سرعت را دو برابر کنیم، این است که **از هر دو نمونه متوالی در ورودی، تنها یک نمونه در خروجی قرار دهیم**. این‌گونه به راحتی سرعت پخش دو برابر خواهد شد و فرکانس نیز به‌ناچار دوبرابر می‌شود. (همانطور که احتمالاً

خودتان هم فکر می‌کنید، نصف اطلاعات در این تبدیل از بین می‌روند! بله! برای همین است که وقتی یک فایل صوتی را با سرعت ۲ برابر پخش می‌کنیم به‌نظر می‌رسد کیفیت کمتری دارد.)

همانند کاری که در بخش reverse انجام داده‌اید، این‌جا نیز باید حواستان به Sample block ها باشد؛ یعنی تغییرات را در سطح block (و نه در سطح بایت یا int یا ..) اعمال کنید.

با توجه به لزوم یکسان بودن خروجی کدها، قرارداد می‌کنیم که Sample block های با اندیس زوج را حفظ کرده و اندیس‌های فرد را حذف می‌کنیم. (یعنی اندیس‌های زوج را به ترتیب داخل فایل جدید می‌ریزیم).

- نوع دوم، `safe`

همانطور که گفتیم، در نوع اول نیمی از داده‌ها از دست می‌رفتند. یک روش دیگر نیز برای تغییر سرعت وجود دارد و آن هم **تغییر `Sample Rate` موجود در هدر فایل است.** در این حالت داده‌های موجود در Data هیچ تغییری نمی‌کنند و صرفاً باید فیلد `Sampling Rate` و همچنین فیلدهای مرتبط (که فرمول‌های آن‌ها نیز در بخش جدول ارائه شده بود) به‌صورت تغییریافته در هدر فایل جدید نوشته شود.

(پی‌نوشت: این حالت صرفاً Player را مجبور می‌کند که با سرعت بالاتری داده‌های صوتی را پخش کند! و با توجه به اینکه پشتیبانی Player ها از Sampling Rate های بالاتر محدودتر است، احتمالا نمی‌توانید فایل خروجی که چندبار این کار را روی آن انجام داده‌اید را پخش کنید!)

- توجه

- در بخش اول توجه کنید که با توجه به کاهش تعداد نمونه‌های مقصد، باید هدر فایل مقصد (فیلدهای `Chunk Size` و `Subchunk 2 Size`) به‌صورت صحیح و به‌روز ذخیره شود.
- در بخش دوم نیز توجه کنید که فیلدهای مرتبط به Sampling Rate (فیلدهای `Byte Rate` و `Sample Rate`) به‌صورت صحیح و به‌روز ذخیره شوند.

توجه!

شما در این تمرین ملزم به پیاده‌سازی کد از صفر نیستید و فایل زیپ شامل تمپلیت پیاده‌سازی، تست‌کیس‌ها و داوری لوکال را می‌توانید از این لینک دانلود نمایید.

مواردی که باید پیاده‌سازی شوند، با TODO درون کد مشخص شده‌اند.

البته داوری لوکال صرفاً برای بررسی توسط خودتان است و در نهایت باید فایل main.c خود را در کوئرا بارگذاری کنید و نمره در کوئرا ملاک خواهد بود.