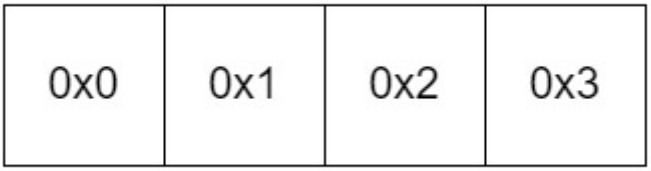
# هاک آقای آبدماغ

- محدودیت زمان: ۱۰۰ میلی‌ثانیه
- محدودیت حافظه: ۱ مگابایت

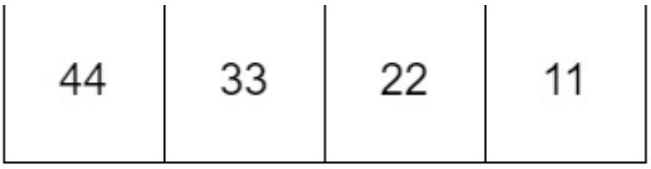
**Endianness:**

برای حل این سوال نیاز به آشنایی با مفهوم endianness دارید. میتوانید به این لینک مراجعه کنید، یا توضیحاتی که در ادامه داده میشود را بخوانید:

کامپیوترها میتوانند به دو دسته little endian و big endian تقسیم شوند. تفاوت این دو دسته در نحوه چینش بایت‌های یک datatype چند بایتی در حافظه است. اکثر datatype هایی که با آنها کار کرده اید مثل integer و float بیشتر از یک بایت جا میگیرند. مثلا هر integer بطور نرمال 4 بایت جا میگیرد. وقتی کامپیوتر میخواهد این 4 بایت را در مموری ذخیره کند، ابتدا یک جایگاه 4 بایتی را برای آن خالی میکند. به یاد آورید که در مموری، هر بایت یک آدرس مخصوص به خودش را دارد و 4 بایت تخصیص داده شده به یک integer همسایه هستند، یعنی آدرس های متوالی دارند. میتوان با این شکل ساختار بلوک حافظه تخصیص داده شده را نشان داد:

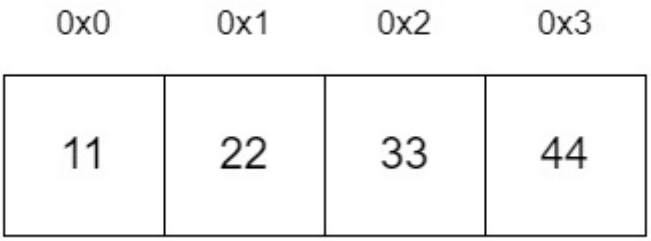



حال فرض کنید یک دیتاتایپ مانند integer داریم که شامل 4 بایت است. بخاطر آورید که هر بایت را میتوان با دو رفتم hexadecimal (مبنای 16) نشان داد. فرض کنید بایت های این اینتجر به صورت $11223344$ هستند. رفتار ماشین little endian به این صورت است که پرارزش ترین بایت (11) را در خانه حافظه با عدد آدرس بیشتر (0x3) قرار میدهد. هرچه ارزش بایت بیشتر باشد، در خانه حافظه‌ای با آدرس بالاتر قرار میگیرد. یعنی در یک دستگاه little endian، عدد گفته شده به این صورت نوشته میشود:




بدیهتا دستگاه endian هنگام خواندن مقدار دیتاتایپ مربوطه، بایت جایگذاری شده در آدرس بالاتر حافظه را با ارزش بیشتر میداند. یعنی به همان صورت $11223344$ این integer را میخواند.

دستگاه‌های big endian اما برعکس دستگاه‌های little endian، بایت پرارزش‌تر را در بایت با آدرس حافظه کوچک‌تر قرار میدهند. یعنی عدد مثال ما را به این صورت ذخیره میکنند:




طبعا این دستگاه‌ها هنگام خواندن مقدار داده‌ساختارهای ذخیره شده، بایت‌هایی که در آدرس‌های کوچک‌تر ذخیره شده‌اند را در جایگاه‌های پرارزش‌تر میخوانند. یعنی عدد ما را به درستی به صورت $11223344$ میخوانند.

*مسئله اصلی:*

▼

داستان مسئله: محتوای این قسمت برای حل سوال نیاز نیست اما در صورت علاقه میتوانید برای خواندن این بلوک را باز کنید.

سالیوان و ووزاوسکی دری که به اتاق Boo میرسید را گم کرده‌اند. برای اینکه محل دقیق آن را پیدا کنند، نیاز دارند وارد سیستم‌های کامپیوتر شرکت هیولاها شوند. سالیوان به کامپیوترهای این شرکت دسترسی فیزیکی دارد، اما برای استفاده از آنها نیازمند پسووردی است که روی لپ‌تاپ شخصی آقای آب‌دماغ ذخیره شده است.

ووزاوسکی موفق میشود شبانه وارد دفتر آقای آب‌دماغ شده و وارد لپ‌تاپش شود و برنامه password manager او را ران کند. این کار باعث شده که پسووردهای شرکت روی مموری لود شوند. حال کافیست

او محتوای مموری را بخواند و پسوورد را بدست آورد!

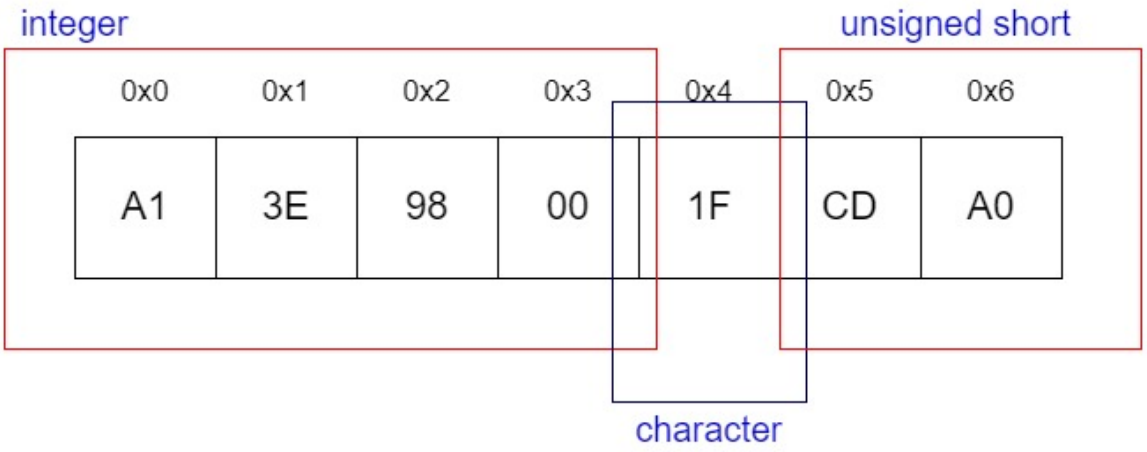
اما مایک متوجه میشود که با دستگاه اشتباه و در datatype های اشتباهی محتوای حافظه را خوانده است (در صورت سوال توضیح داده میشود) و حالا شما باید برنامه‌ای بنویسید که طبق اطلاعاتی که مایک در اختیارتان قرار میدهد مقدار درست بخش‌های پسوورد را بدست آورد.

ووزاوسکی توانسته به بخشی از مموری کامپیوتر آقای آب‌دماغ که پسوورد سرورها رویش نوشته شده دست یابد. این تکه مموری n بایت طول دارد و این n بایت در مجاورت هم قرار دارند. در این n بایت یک سری datatype نوشته شده‌اند که محتوای آنها بخش‌های مختلف رمز سرور اصلی را دربر میگرد. مایک محتوای تکه حافظه‌ی مذکور را به یک دستگاه big endian داده تا بخواند. متاسفانه از قبل نمیدانیم محتوای مموری چه است، اما فرض کنید بلوک مموری به این صورت باشد (شامل 7 بایت):

| 0x0 | 0x1 | 0x2 | 0x3 | 0x4 | 0x5 | 0x6 |
|---|---|---|---|---|---|---|
| A1 | 3E | 98 | 00 | 1F | CD | A0 |

مایکی با دستگاه big endian اش توانسته این 7 بایت را به صورت یک integer (۴ بایت اول)، یک کاراکتر (۱ بایت بعدی) و سپس یک unsigned short (2 بایت آخر) بخواند. ترتیب تخصیص قسمت‌های

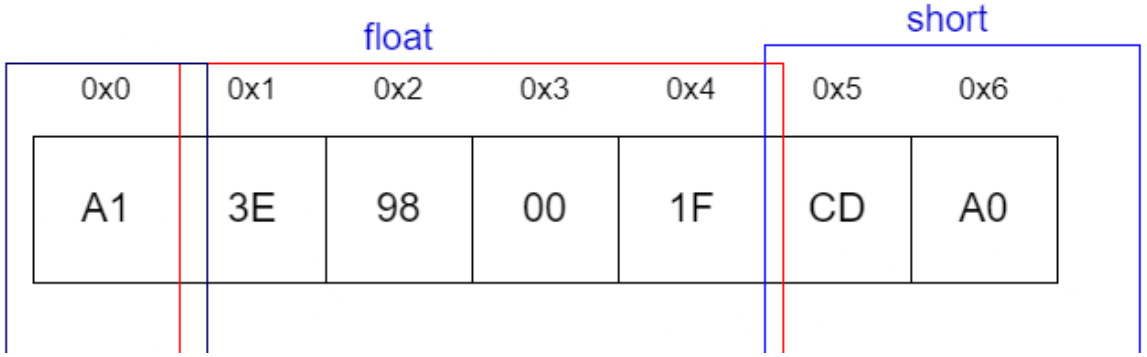
این بلوک از کوچک به بزرگ طبق آدرس حافظه است. یعنی ۴ بایتی که در قسمت حافظه با آدرس های کمتر قرار دارند را بعنوان ۱ ،integer بایت بعدی را بعنوان کاراکتر، و ۲ بایت با آدرس بزرگ تر را بعنوان unsigned short خوانده است. در تصویر میتوانید نحوه تقسیم را ببینید:

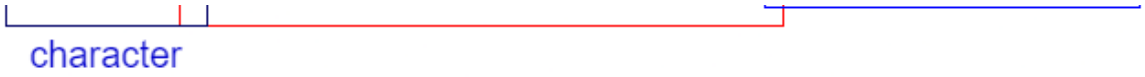



آدرس‌های ۰ تا ۳ مربوط به اینتجر، آدرس ۴ مربوط به کاراکتر، و آدرس‌های ۵ و ۶ مربوط به unsigned short شده اند. دقت کنید که هرکدام از این داده ساختارها به صورت big endian خوانده شده اند، یعنی مثلا بایت‌های اینتجر طوری خوانده شده‌اند که مقدار hex آنها به این صورت است:

$$A13E9800$$

ووزاوسکی توانسته به صورت گفته شده محتوای حافظه را بخواند و بعنوان ورودی آن را به برنامه شما خواهد داد. شما از طریق دیگری (لاگ‌های روی دیسک آقای آب‌دماغ که مایک با خود برایتان آورده) متوجه می‌شوید که در واقع datatypeهای دیگری در آن قسمت‌های حافظه ذخیره شده بوده‌اند. مثلا می‌فهمید در واقع به ترتیب یک char، سپس یک float و بعد یک short (از نوع علامتدار) در آن خانه‌های حافظه ذخیره شده بودند. یعنی در واقع تقسیم بندی قسمت‌های حافظه به صورت زیر بوده:






علاوه بر این، شما متوجه میشوید که اصلا کامپیوتر آقای آب‌دماغ از نوع little endian بوده است و مقادیر ذخیره شده روی حافظه‌ای که فرد مذکور خوانده بوده، در اصل بصورت little endian نوشته شده بوده اند. یعنی اگر بایت‌های float مذکور را به ترتیب درست کنار هم بچینیم، مقدار hexadecimal آن‌ها به این صورت خواهد شد:

$$1F00983E$$

همچنین می‌دانید که کد شما قرار است حتما روی ماشینی که little endian است اجرا شود. حال شما باید مقادیر درست datatype هایی که روی کامپیوتر اصلی نوشته شده بوده است را پیدا کنید. یعنی مثلا در این مورد ابتدا مقدار $A1$ را بصورت یک کاراکتر پرینت کنید، سپس یک فلوت با مقدار داده شده پرینت کنید، و سپس یک short با مقدار مناسب پرینت کنید (یعنی مقداری که در کامپیوتر اصلی ذخیره شده بوده است.)

## ورودی

در ورودی ابتدا تعداد datatype هایی که فرد مذکور از روی حافظه خوانده است و سپس در خطوط بعدی کد خود دیتاتایپ‌ها روی خطهای مجزا داده میشوند.

تایپ‌ها محدود به مجموعه‌ی زیرند و کدهای آنها روبرویشان نوشته شده است: البته در این قسمت تضمین میشود که float و double داده نمیشود.

| datatype | code | size (bytes) |
|---|---|---|
| int | 0 | 4 |
| unsigned int | 1 | 4 |
| short | 2 | 2 |
| unsigned short | 3 | 2 |
| char | 4 | 1 |

| datatype | code | size (bytes) |
|---|---|---|
| long long | 5 | 8 |
| float | 6 | 4 |
| double | 7 | 8 |

مثلا در مثال زده شده در صورت سوال، این مقادیر داده خواهند شد:

```
3
0
4
3
```

دیده می‌شود در خط اول تعداد تایپ‌ها (۳) و سپس کد تایپ‌ها (اینتجر و کاراکتر و unsigned short) داده شده است.

سپس مقادیر خوانده شده توسط فرد مذکور برای قسمت مورد نظر از حافظه، به ترتیبی که تایپ‌هایشان داده شده‌اند و در خطوط مجزا ورودی داده می‌شوند. مثلا در مثال زده شده در صورت سوال ممکن است چنین ورودی‌ای داده شود:

```
-1235980325
c
235
```

سپس تعداد تایپ‌هایی که در اصل روی حافظه نوشته شده بوده اند، و در ادامه روی خط‌های مجزا کدهای آن تایپ‌ها داده میشوند. تضمین می‌شود مجموع اندازه‌های تایپ‌های سری اول (آنهایی که از روی حافظه خوانده شده اند) و تایپ‌های سری دوم (آنهایی که شما باید چاپ کنید) یکسان خواهد بود. دقت کنید در این قسمت ممکن است تایپ double یا float داده شود.

در مثال زده شده در صورت سوال این خطوط داده خواهند شد:

```
3
4
6
```

```
2
```

## خروجی

در خروجی شما باید به ازای هر تایپی که در لپ‌تاپ آقای آب‌دماغ نوشته شده بود (دسته‌ی دوم تایپ‌های داده شده در ورودی سوال) مقدار درست آن را پرینت کنید. مثلا در مثال زده شده باید در خطوط مجزا ابتدا یک کاراکتر، سپس یک فلوت، و در نهایت یک short پرینت کنید.

**نکته مهم: فلوت‌ها را بصورت scientific notation (با e%) و با 2 رقم اعشار، و double ها را نیز بصورت scientific notation ولی با ۴ رقم اعشار پرینت کنید.**

## مثال

### ورودی نمونه ۱

مثالی که در صورت سوال دنبال شد به این صورت خواهد بود:

```
3
0
4
3
1242579194
B
255
3
4
6
2
```

### خروجی نمونه 1

```
J
1.25e+02
-256
```