شهر زیرزمینی تمساح‌ها (Crocodile's Underground City)

بنیاماس(Benjamas) یک باستان‌شناس است که پس از کشف شهر زیرزمینی تمساح‌ها برای نجات دادن جان خود در تلاش است. این شهر شامل N اتاق است. M راهروی دو طرفه نیز وجود دارد که هر کدام از راهروها دو اتاق متفاوت را به یکدیگر وصل می‌کنند. گذر از راهروهای متفاوت ممکن است زمان‌های متفاوتی طول بکشد. تنها K تا از این N اتاق، اتاق خروجی هستند که به بنیاماس اجازه‌ی فرار می‌دهند. بنیاماس اکنون در اتاق شماره‌ی صفر هست و می‌خواهد هر چه سریع‌تر خود را به یکی از اتاق‌های خروجی برساند.

نگهبان تمساح‌ها می‌خواهد مانع فرار بنیاماس شود. او از اتاق خودش درب‌های مخفی‌ای را کنترل می‌کند که با کمک آنها در هر لحظه تنها **یک راهرو** را می‌تواند مسدود کند. به‌این معنی‌که هر گاه او یک راهروی جدید را مسدود می‌کند، راهرویی که قبلاً مسدود شده باز می‌شود.

وضعیت بنیاماس را می‌توان به این شکل توضیح داد: هر زمانی که او تلاش می‌کند از یک اتاق خارج شود، نگهبان ممکن است یکی از راهروهای مجاور آن اتاق را مسدود کند. لذا، بنیاماس از طریق یکی از راهروهای مسدود نشده به اتاق دیگری می‌رود. وقتی بنیاماس وارد یک راهرو می‌شود، نگهبان نمی‌تواند تا زمانی که بنیامین به انتهای دیگر راهرو نرسیده است، آن راهرو را مسدود کند. در زمانی که بنیاماس به اتاق جدید رسید، نگهبان می‌تواند راهرو جدیدی مجاور اتاق جدید (حتی راهرویی که اخیراً عبور شده) را مسدود کند و ... .

بنیاماس مایل است که یک برنامه‌ی فرارِ آسان از قبل داشته باشد. به طور دقیق‌تر، او علاقه‌مند است که مجموعه دستورالعملی داشته باشد که به او بگوید که بعد از این که وارد یک اتاق شد چه کند. فرض کنید A یکی از اتاق‌ها باشد. بدیهی است که اگر A یک اتاق خروجی باشد نیازی به هیچ دستورالعملی نیست چرا که او می‌تواند از طریق همین اتاق از شهر فرار کند. در غیر این صورت، دستورالعمل مربوط به اتاق A باید به یکی از دو صورت زیر باشد:

- «اگر به اتاق A رسیدی، راهروی متصل به اتاق B را طی کن. ولی اگر این راهرو مسدود بود، راهروی متصل به اتاق C را طی کن.»
- «نگران اتاق A نباش. طبق برنامه‌ی فرار، هیچ‌گاه به این اتاق نخواهی رسید.»

توجه کنید که در بعضی حالات (برای مثال، اگر برنامه شما بنیاماس را داخل یک حلقه بیندازد) نگهبان ممکن است قادر به جلوگیری از رسیدن بنیاماس به اتاق‌های خروج شود. یک برنامه‌ی فرار، **خوب** است اگر تضمین کند که مستقل از نحوه‌ی کار نگهبان، بنیاماس را در زمانی متناهی به یک اتاق خروج برساند. برای یک برنامه‌ی فرار خوب، فرض کنید T کمترین زمانی باشد که برنامه تضمین می‌کند همواره بنیاماس (مستقل از عمل‌کرد نگهبان) در آن زمان به یک اتاق خروجی برسد. در این حالت می‌گوییم این برنامه‌ی فرارِ خوب، T واحد زمان طول می‌کشد.

تمساح

فارسی نسخه ۱٫۳

وظیفه شما

شما باید روال **travel_plan(*N*, *M*, *R*, *L*, *K*, *P*)** را بنویسید که پارامترهای زیر را بگیرد:

- ***N*** : تعداد اتاق‌ها. اتاق‌ها با شماره‌ی صفر تا $N-1$ شماره‌گذاری شده‌اند.
- ***M*** : تعداد راهروها. راهروها با شماره‌ی صفر تا $M-1$ شماره‌گذاری شده‌اند.
- ***R*** : یک آرایه‌ی دوبُعدی از اعداد صحیح که راهروها را نمایش می‌دهد. برای $0 \le i < M$، راهروی i دو اتاق متمایز $R[i][0]$ و $R[i][1]$ را به هم وصل می‌کند. هر دو اتاق با حداکثر یک راهرو به یکدیگر وصل می‌شوند.
- ***L*** : یک آرایه‌ی یک‌بُعدی از اعداد صحیح که شامل زمان گذر از هر راهرو است. برای $0 \le i < M$، زمان $1 \le L[i] < 1{,}000{,}000{,}000$ زمانیست که بنیاماس نیاز دارد که از اتاق $R[i][0]$ به اتاق $R[i][1]$ برسد.
- ***K*** : تعداد اتاق‌های خروجی. می‌توانید فرض کنید که $1 \le K < N$.
- ***P***: یک آرایه‌ی یک‌بُعدی با K عدد متفاوت شامل شماره‌ی اتاق‌های خروجی. برای $0 \le i < K$، مقدار $P[i]$، شماره‌ی اتاق خروجی i-اُم است. اتاق شماره صفر هیچ‌گاه یک اتاق خروجی نخواهد بود.

روال شما می‌بایست کمترین زمان T که برای آن، یک برنامه‌ی فرار خوب وجود دارد را برگرداند.

می‌توانید فرض کنید که هر اتاقی که خروجی نیست حداقل دو راهرو متصل به خود دارد. هم‌چنین می‌توانید فرض کنید که برای هر ورودی یک برنامه فرار خوب با زمان $T \le 1{,}000{,}000{,}000$ موجود است.

مثال نمونه

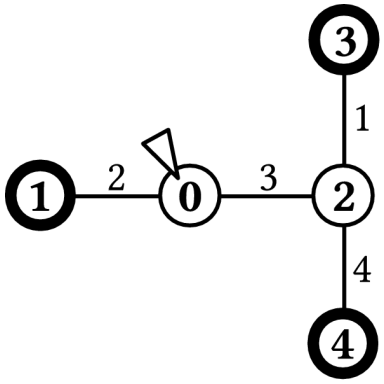
مثال ۱

حالتی را در نظر بگیرید که $N = 5$، $M = 4$، $K = 3$ و

R=
0 1
0 2
3 2
2 4

L=
2
3
1
4

P=
1
3
4



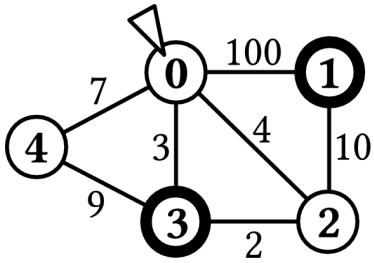
در شکل بالا، اتاق‌ها با دایره و راهروها با پاره‌خط نمایش داده شده‌اند. اتاق‌های خروجی با دایره‌های ضخیم نمایش داده شده‌اند. بنیاماس از اتاق صفر (که با مثلث مشخص شده) شروع می‌کند. یک برنامه‌ی فرار بهینه به این شکل است:

- هرگاه در اتاق صفر بودی، راهرویی را برو که به اتاق ۱ می‌رود. اگر آن راهرو مسدود بود، راهرویی را برو که به اتاق ۲ می‌رود.
- اگر در اتاق ۲ بودی، راهرویی را برو که به اتاق ۳ می‌رود. اگر آن راهرو مسدود بود، راهرویی را برو که به اتاق ۴ می‌رود.

در بدترین حالت، بنیاماس بعد از ۷ واحد زمانی به یکی از اتاق‌های خروجی می‌رسد. لذا `travel_plan` باید مقدار ۷ را برگرداند.

مثال ۲

حالتی را در نظر بگیرید که $N = ۵$، $M = ۷$ و $K = ۲$



$$R = \begin{matrix} 0 & 2 \\ 0 & 3 \\ 3 & 2 \\ 2 & 1 \\ 0 & 1 \\ 0 & 4 \\ 3 & 4 \end{matrix} \quad L = \begin{matrix} 4 \\ 3 \\ 2 \\ 10 \\ 100 \\ 7 \\ 9 \end{matrix} \quad P = \begin{matrix} 1 \\ 3 \end{matrix}$$

یک برنامه‌ی فرار بهینه به این صورت است:

- هرگاه در اتاق صفر بودی به راهرویی برو که به اتاق ۳ می‌رود و اگر آن راهرو مسدود بود به راهرویی برو که به اتاق ۲ می‌رود.
- هرگاه در اتاق ۲ بودی به راهرویی برو که به اتاق ۳ می‌رود و اگر آن راهرو مسدود بود به راهرویی برو که به اتاق ۱ می‌رود.
- نگران اتاق ۴ نباش. بر اساس این برنامه، هیچ‌گاه به آن نمی‌رسی.

بنیاماس بعد از حداکثر ۱۴ واحد زمان، حتماً به یکی از اتاق‌های خروجی می‌رسد. پس `travel_plan` باید ۱۴ را برگرداند.

زیرمسئله‌ها

زیرمسئله شماره یک (۴۶ امتیاز)

- $۳ \le N \le ۱٫۰۰۰$
- شهر زیرزمینی یک درخت است. یعنی، $M = N - ۱$ و بین هر دو اتاق یک مسیر وجود دارد که آن‌ها را به هم وصل می‌کند.
- هر اتاق خروجی دقیقا به یک اتاق دیگر وصل است.
- هر اتاق دیگر حداقل به ۳ اتاق دیگر وصل است.

زیرمسئله شماره دو (۴۳ امتیاز)

- $۳ \le N \le ۱٫۰۰۰$
- $۲ \le M \le ۱۰۰٫۰۰۰$

زیرمسئله شماره سه (۱۱ امتیاز)

- $۳ \le N \le ۱۰۰٫۰۰۰$
- $۲ \le M \le ۱٫۰۰۰٫۰۰۰$

جزئیات پیاده‌سازی

محدودیت‌ها

- **محدودیت زمانی CPU:** دو ثانیه
- **محدودیت حافظه:** ۲۵۶ مگابایت
- **توجه:** هیچ محدودیت صریحی روی اندازه‌ی حافظه‌ی پشته (stack) استفاده شده وجود ندارد. میزان حافظه‌ی استفاده شده در پشته به‌عنوان بخشی از حافظه‌ی مصرفی محاسبه می‌شود.

واسط API

- فولدرِ پیاده‌سازی: /crocodile
- فایل‌هایی که توسط شرکت‌کننده می‌بایست پیاده‌سازی شوند: crocodile.c یا crocodile.cpp یا crocodile.pas
- واسط شرکت‌کننده: crocodile.h یا crocodile.pas
- واسط مصحح: crocodile.h یا crocodilelib.pas
- مصحّحِ نمونه: grader.c یا grader.cpp یا grader.pas و crocodilelib.pas
- ورودی مصحّح نمونه: grader.in.1، grader.in.2 و ...
- **توجه:** مصحح نمونه، ورودی را به فرمت زیر می‌خواند:
  - **خط اوّل:** مقادیر N، M و در نهایت K.
  - **خط ۲ الی $M+1$ اُم:** برای $0 \le i < M$، خط $i+2$ شامل $R[i][0]$ و $R[i][1]$ و $L[i]$ می‌باشد که با یک فاصله از هم جدا شده‌اند.
  - **خط $M+2$ اُم:** یک لیست شامل K عدد صحیح $P[K-1], \cdots, P[1], P[0]$ که با یک فاصله از هم جدا شده‌اند.
  - **خط $M+3$:** جواب مورد انتظار.
- خروجی مورد انتظار برای ورودی مصحح نمونه: grader.expect.1، grader.expect.2 و ...

برای این مسئله، هر کدام از این فایل‌ها دقیقاً باید شامل عبارت «**Correct.**» باشند.

المپیاد جهانی کامپیوتر سال ۲۰۱۱
۲۲ الی ۲۹ ژولای ۲۰۱۱، شهر پاتایا، تایلند
مسائل مسابقه – روز دوم

فیل‌های رقصنده

«فیل‌های رقصنده» یک نمایش (show) پربیننده در پاتایا است که در آن N فیل روی یک خط (که به آن صحنه می‌گویند) می‌رقصند.

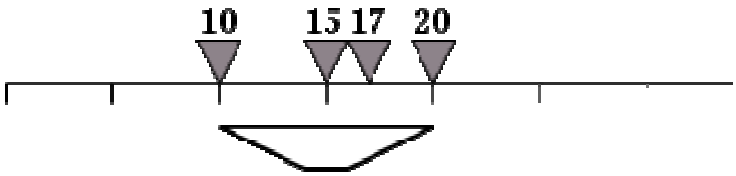
پس از سال‌ها آموزش، فیل‌های این نمایش قادر به انجام رقص‌های جالبی هستند. نمایش شامل تعدادی اجرا (act) است که در هر اجرا، دقیقاً یک فیل رقص زیبایی انجام می‌دهد و در این حین ممکن است مکان خود را عوض کند.

تهیه‌کنندگان نمایش می‌خواهند کتاب عکسی تولید کنند که عکس‌های کلّ نمایش را داشته باشد. بعد از هر اجرا، آن‌ها می‌خواهند عکس‌هایی از همه‌ی فیل‌ها، به همان شکلی که بینندگان می‌بینند، بگیرند.

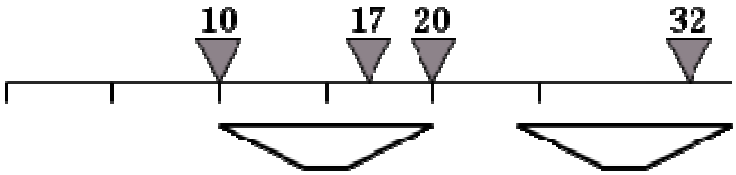
در هر زمانی از نمایش، تعدادی فیل ممکن است مکان یکسانی داشته باشند. در این صورت، آن‌ها پشت هم در همان‌مکان قرار می‌گیرند.

یک دوربین می‌تواند یک عکس از یک گروه از فیل‌ها بگیرد اگر و تنها اگر مکان آن گروه از فیل‌ها روی پاره‌خطی به طول L (شامل هر دو نقطه‌ی دو سر پاره‌خط) جا شود. از آن‌جا که فیل‌ها می‌توانند روی صحنه پخش شوند ممکن است به چندین دوربین برای ثبت هم‌زمان تصویر همه فیل‌ها نیاز داشته باشیم.

برای مثال، فرض کنید $L = ۱۰$ و فیل‌ها در مکان‌های ۱۰، ۱۵، ۱۷ و ۲۰ صحنه قرار دارند. در این لحظه، یک دوربین (همان‌طور که در شکل زیر می‌بینید) می‌تواند تصویرشان را بگیرد. (فیل‌ها در شکل به صورت مثلث و دوربین‌ها با ذوزنقه نمایش داده شده‌اند.)



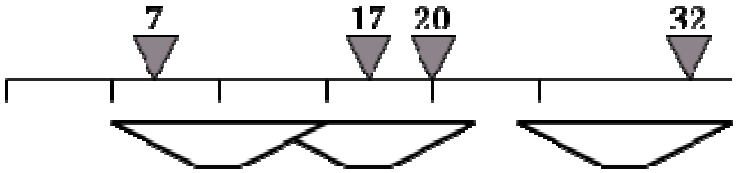
در اجرای بعدی، فیلی که در مکان ۱۵ هست رقص‌کنان به مکان ۳۲ می‌رود. بعد از این اجرا، ما نیاز به حداقل ۲ دوربین برای ثبت تصویر داریم.



در اجرای بعدی، فیلی که در مکان ۱۰ است به مکان ۷ می‌رود. در این وضعیت جدید فیل‌ها، ما نیاز به ۳ دوربین برای عکس گرفتن از همه فیل‌ها داریم.

فیل‌ها

فارسی نسخه ۱٫۳



در این مسئله‌ی تعاملی (Interactive)، شما باید **کمترین** تعداد دوربین مورد نیاز برای تصویربرداری بعد از هر اجرا را معین کنید. توجه کنید که بین هر دو اجرای متوالی، تعداد دوربین‌های مورد نیاز ممکن است زیاد یا کم شود و یا تغییری نکند.

وظیفه شما

شما باید روال‌های زیر را پیاده‌سازی کنید.

- روال **`init(N, L, X)`** که پارامترهای زیر را می‌گیرد:
  - N : تعداد فیل‌ها که با شماره‌های صفر تا $N-1$ شماره‌گذاری شده‌اند.
  - L : طول پاره‌خطی است که با یک دوربین به تنهایی ضبط می‌شود. می‌توانید فرض کنید که $0 \le L \le 1{,}000{,}000{,}000$ است.
  - X : یک آرایه‌ی یک‌بُعدی از اعداد صحیح که مکان آغازین فیل‌ها را مشخص می‌کند. برای $0 \le i < N$، مکان آغازین فیل i-اُم برابر با $X[i]$ است. مکان‌های آغازین به صورت صعودی مرتب‌شده‌اند. یعنی می‌توانید فرض کنید که $0 \le X[0] \le \cdots \le X[N-1] \le 1{,}000{,}000{,}000$ است. توجه داشته باشید که در حین رقص، فیل‌ها ممکن است مکان‌شان را تغییر دهند.

این روال فقط یک بار (در ابتدای کار و پیش از همه‌ی فراخوانی‌های روال `update`) صدا زده خواهد شد و هیچ مقداری را برنمی‌گرداند.

- روال **`update(i, y)`** که پارامترهای زیر را می‌گیرد:
  - i : شماره‌ی فیلی که در اجرای فعلی حرکت می‌کند.
  - y : مکان فیل i بعد از اجرا. می‌توانید فرض کنید که $0 \le y \le 1{,}000{,}000{,}000$ است و y یک عدد صحیح است.

این روال چندین بار فراخوانی می‌شود. هر فراخوانی معادل یک اجرا است (که بعد از اجراهای قبلی انجام می‌شود). هر فراخوانی باید **کمترین تعداد** دوربین مورد نیاز برای تصویربرداری از همه‌ی فیل‌ها بعد از اجرای متناظر را برگرداند.

مثال نمونه

مثال اوّل

حالتی را در نظر بگیرید که $N = ۴$، $L = ۱۰$ و مکان آغازین فیل‌ها به صورت زیر است:

$$X = \begin{matrix} 10 \\ 15 \\ 17 \\ 20 \end{matrix}$$

ابتدا، روال `init` شما با این پارامترها فراخوانی خواهد شد. پس از آن، روال `update` شما یک‌بار برای هر اجرا فراخوانی می‌شود.

در مثال زیر، دنباله‌ای از فراخوانی‌ها و مقدار صحیح برگشتی آن‌ها آمده است:

شماره اجرا	پارامترهای ارسالی	مقدار برگشتی
۱	`update(2,16)`	1
۲	`update(1,25)`	2
۳	`update(3,35)`	2
۴	`update(0,38)`	2
۵	`update(2,0)`	3

زیرمسئله‌ها

زیرمسئله شماره یک (۱۰ امتیاز)

- دقیقاً $N = ۲$ فیل وجود دارند.
- در ابتدا و بعد از هر اجرا، مکان فیل‌ها متفاوت است.
- روال `update` شما حداکثر **۱۰۰** بار فراخوانی می‌شود.

زیرمسئله شماره دو (۱۶ امتیاز)

- $۱ \le N \le ۱۰۰$
- در ابتدا و بعد از هر اجرا، مکان فیل‌ها متفاوت است.
- روال `update` شما حداکثر **۱۰۰** بار فراخوانی می‌شود.

زیرمسئله شماره سه (۲۴ امتیاز)

- $۱ \le N \le ۵۰٫۰۰۰$
- در ابتدا و بعد از هر اجرا، مکان فیل‌ها متفاوت است.
- روال `update` شما حداکثر **۵۰٫۰۰۰** بار فراخوانی می‌شود.

زیرمسئله شماره چهار (۴۷ امتیاز)

- $۱ \le N \le ۷۰٫۰۰۰$
- در یک مکان ممکن است بیشتر از یک فیل قرار گیرد.
- روال `update` شما حداکثر **۷۰٫۰۰۰** بار فراخوانی می‌شود.

زیرمسئله شماره پنج (۳ امتیاز)

- $۱ \le N \le ۱۵۰٫۰۰۰$
- در یک مکان ممکن است بیشتر از یک فیل قرار گیرد.
- روال `update` شما حداکثر **۱۵۰٫۰۰۰** بار فراخوانی می‌شود.
- نکته مربوط به محدودیت زمانی CPU را در قسمت جزئیات پیاده‌سازی ببینید.

جزئیات پیاده‌سازی

محدودیت‌ها

- **محدودیت زمانی CPU:** ۹ ثانیه

توجه: collection templateها در کتابخانه استاندارد C++ (STL) ممکن است کند باشند. مخصوصاً، اگر از آن‌ها استفاده کنید، ممکن است نتوانید زیر مسئله ۵ را حل کنید.

- **محدودیت حافظه:** ۲۵۶ مگابایت
- **توجه:** هیچ محدودیت صریحی روی اندازه‌ی حافظه‌ی پشته (stack) استفاده شده وجود ندارد. میزان حافظه‌ی استفاده شده در پشته به‌عنوان بخشی از حافظه‌ی مصرفی محاسبه می‌شود.

واسط API

- فولدرِ پیاده‌سازی: elephants/
- فایل‌هایی که توسط شرکت‌کننده می‌بایست پیاده‌سازی شوند: elephants.c یا elephants.cpp یا elephants.pas
- واسط شرکت‌کننده: elephants.h یا elephants.pas
- مصحّحِ نمونه: grader.c یا grader.cpp یا grader.pas
- ورودی مصحّح نمونه: grader.in.1، grader.in.2 و ...
- **توجه:** مصحح نمونه، ورودی را به فرمت زیر می‌خواند:
  - **خط اوّل:** مقادیر N، L و در نهایت M، که M تعداد اجراها در نمایش است.
  - **خط‌های ۲ الی $N+1$ اُم:** مکان‌های آغازین. یعنی خط $K+2$ (برای $0 \le K < N$) شامل مقدار $X[K]$ است.
  - **خط‌های $N+2$ تا $N+M+1$اُم:** اطلاعات مربوط به M اجرا. یعنی برای هر $1 \le j \le M$، خط $N+1+j$ شامل سه مقدار $i[j]$، $y[j]$، و $s[j]$ که با یک فاصله از هم جدا شده‌اند به این معنی که در j-امین اجرا، فیل شماره $i[j]$ به مکان $y[j]$ می‌رود و بعد از آن اجرا، $s[j]$ کم‌ترین تعداد دوربین مورد نیاز برای تصویربرداریست.
- خروجی مورد انتظار برای ورودی مصحح نمونه: grader.expect.1، grader.expect.2 و ...

برای این مسئله، هر کدام از این فایل‌ها دقیقاً باید شامل عبارت «**Correct.**» باشند.

کفترها (Parrots)

یانی (Yanee) یک کفترباز است. از زمانی که یانی راجع به سیستم IPoAC (مخفف «انتقال اطلاعات با استفاده از پرندگان») چیزهایی شنیده است مصمم شده تا یک گروه از کبوتران هوشمند را برای انتقال پیام‌هایی به مقصدهای دوردست تربیت کند.

رؤیای یانی این است که بتواند از پرندگانش برای ارسال یک پیام M به یک سرزمین خیلی خیلی دور استفاده کند. پیام M که یانی قصد ارسال آن را دارد، از N عدد صحیح (نه لزوماً متفاوت) در محدوده‌ی صفر تا ۲۵۵ (شامل این دو عدد) تشکیل شده است. یانی K عدد کفتر که به‌صورت ویژه تعلیم داده شده‌اند، برای این منظور دارد. منتهی تمام پرنده‌ها کاملاً شبیه هم هستند و یانی نمی‌تواند بین آن‌ها تمایزی قائل شود. هر کدام از این کفترها هم، تنها یک عدد صحیح بین صفر تا R (شامل این دو عدد) را می‌تواند به‌خاطر سپرده و حمل کند.

در ابتدا یانی یک مدل بسیار ساده را پیاده‌سازی کرد: او برای ارسال پیام، پرنده‌ها را یک به یک از قفس‌شان بیرون آورده و به هر یک از آن‌ها یک عدد از دنباله را به‌ترتیب آموزش داد. سپس کفتران را به پرواز در آورد. بدبختانه، این روش جواب نداد؛ چرا که، پرندگان الزاماً به‌همان ترتیبی که پرواز را شروع کرده بودند، به مقصد نرسیدند. نتیجتاً در این روش، یانی می‌توانست تمامی اعداد را بازیابی کند، منتهی نمی‌توانست ترتیب درست بین این اعداد را بیابد.

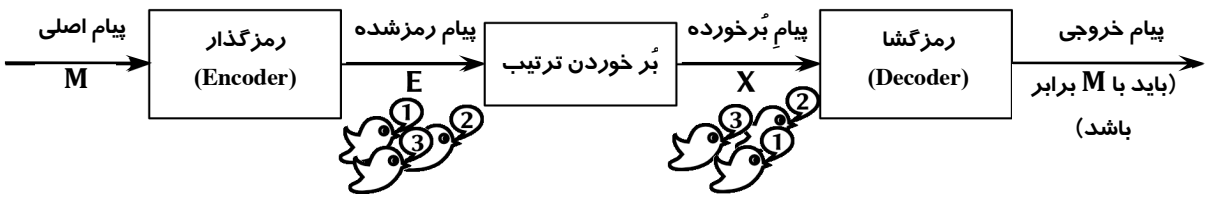
برای تحقّق رؤیای یانی، او به مدل بهتری نیاز دارد و به همین دلیل از شما کمک می‌خواهد. با داشتن پیام M، یانی همواره ابتدا کفتران را یکی یکی از قفس درآورده و مشابه روش قبل پس از آموختن یک عدد به هر کدام‌شان، آن‌ها را به پرواز در می‌آورد. یانی از شما تقاضا دارد که برنامه‌ای را بنویسید که دو عمل مجزای زیر را انجام دهد:

- به‌عنوان عمل اوّل، برنامه شما باید یک پیام M را خوانده و آن را به دنباله‌ای از حداکثر K تا عددِ صحیح بین صفر تا R (شامل این دو عدد) تبدیل کند. یانی این اعداد را به پرندگان خواهد آموخت.
- به‌عنوان عمل دوم، برنامه شما باید یک لیست از اعداد بین صفر تا R (شامل این دو عدد) را که اعدادی هستند که کفتران به مقصد رسانده‌اند، دریافت کرده و پس از تحلیل، پیام اصلی M را از آن استخراج کند.

می‌توانید فرض کنید که تمام کفتران همیشه به مقصد می‌رسند و هر کدام از آن‌ها هم عدد اختصاصی‌اش را به‌درستی به‌خاطر می‌سپارد. یانی یک بار دیگر به شما یادآور می‌شود که کفتران ممکن‌ست به هر ترتیبی به مقصد برسند. توجه داشته باشید که یانی K تا کفتر بیشتر ندارد و از همین رو، دنباله‌ای از اعداد صحیح بین صفر تا R که شما تولید می‌کنید، باید حداکثر شامل K تا عدد باشد.

وظیفه شما

شما باید دو روال مجزّا بنویسید. یکی از آن‌ها به‌عنوان ارسال کننده (رمزگذار یا encoder)، و دیگری به‌عنوان دریافت‌کننده (رمزگشا یا decoder) استفاده می‌شود. کلِّ پروسه در شکل زیر نمایش داده شده است.



دو روالی که شما می‌بایست بنویسید به شرح زیر هستند.

- روال **encode(N, M)** پارامترهای زیر را می‌گیرد:
  - N : طول پیام اصلی
  - M : یک آرایه‌ی یک‌بُعدی از N عدد صحیح که تشکیل‌دهنده‌ی متن پیام هستند. می‌توانید فرض کنید که برای هر $0 \le i < N$ الزاماً $0 \le M[i] \le 255$ می‌باشد.

این روال باید پیام M را به دنباله‌ای از اعداد صحیح بین صفر تا R (شامل این دو عدد) که قرار است توسّط کفتران ارسال شوند، تبدیل کند. برای گزارش کردنِ (report کردن) این دنباله‌ی تبدیل شده، روال encode می‌بایست روال **send(a)** را برای هر عدد **a** ای که می‌خواهید این a به دقیقاً یک کفتر داده شود، یک بار صدا کند.

- روال **decode(N, L, X)** پارامترهای زیر را می‌گیرد:
  - N : طول پیام اصلی
  - L : طول پیام دریافتی (تعداد کفترهایی که ارسال شده‌اند)
  - X : یک آرایه‌ی یک‌بُعدی از L عدد صحیح، که اعداد دریافت‌شده را نشان می‌دهند. اعداد $X[i]$ برای iهای بین $0 \le i < L$، دقیقاً اعدادی هستند که روال encode شما آن‌ها را تولید کرده، منتهی ترتیب‌شان احتمالاً تغییر کرده است.

این روال می بایست پیام اصلی را بازیابی کند. برای گزارش کردنِ این پیامِ بازیابی شده، روال decode می‌بایست روال **output(b)** را برای اعداد صحیح **b** در پیامِ بازیابی شده به‌ترتیب فراخوانی کند.

توجه کنید که R و K به‌عنوان پارامتر ورودی داده نمی‌شوند. توضیحات زیرمسئله‌ها را در ادامه ببینید.

برای این که شما یک زیرمسئله را کامل حل کنید، شروط زیر می‌بایست در مورد روال‌های شما صادق باشد:

- تمامی اعدادی که توسط روال encode شما به سیستم ارسال می‌شوند، می‌بایست در محدوده‌ی تعیین شده‌ی زیرمساله باشند.
- تعداد دفعاتی که روال encode روال send را فرا می‌خواند نباید از حدّ مشخص شده در زیر مسئله (K) بیش‌تر بشود. توجه کنید که K به طول پیام وابسته است.
- روال decode می‌بایست دقیقاً و به‌طور صحیح پیام اصلی M را بازیابی کرده و روال output(b) را دقیقاً N بار برای b مساوی با $M[0]$، $M[1]$، ... و $M[N-1]$ به‌ترتیب صدا بزند.

در آخرین زیرمسئله، نمره‌ی شما وابسته به نسبت طول پیام رمزگذاری شده به طول پیام اصلی است.

مثال نمونه

حالتی را در نظر بگیرید که $N = 3$ بوده و پیام M به‌صورت مقابل باشد.

$$M = \begin{matrix} 10 \\ 30 \\ 20 \end{matrix}$$

فرض کنید در این حالت، روال encode(N, M) با استراتژی ناشناخته‌ای این پیام را به دنباله‌ی $\langle 7,3,2,70,15,20,3 \rangle$ رمزگذاری (encode) کند. برای ارسال این دنباله، برنامه می‌بایست روال send را به‌ترتیب زیر فراخوانی کند:

```
send(7)
send(3)
send(2)
send(70)
send(15)
send(20)
send(3)
```

حال فرض کنید که زمانی‌که تمامی کفتران به مقصد خود رسیدند، ما لیست اعداد را به‌صورت $\langle 3,20,70,15,2,3,7 \rangle$ دریافت می‌کنیم. اکنون روال decode با پارامترهای $N = ۳$ و $L = ۷$ و X به‌صورت روبه‌رو فراخوانی می‌شود:

$$X = \begin{matrix} 3 \\ 20 \\ 70 \\ 15 \\ 2 \\ 3 \\ 7 \end{matrix}$$

روال decode می‌بایست با داشتن این اطلاعات پیام اصلی $\langle 10,30,20 \rangle$ را بسازد. برای گزارش می‌بایست روال output به ترتیب زیر صدا زده شود.

```
output(10)
output(30)
output(20)
```

زیرمسئله‌ها

زیرمسئله‌ی شماره یک (۱۷ امتیاز)

- $N = ۸$ است و هر یک از اعداد آرایه‌ی M یا صفر و یا یک هستند.
- هر عدد رمزگذاری شده باید در محدوده‌ی **صفر الی** $R = ۶۵۵۳۵$ (شامل این دو عدد) باشد.
- تعداد دفعاتی که شما می‌توانید روال send را فراخوانی کنید حداکثر $K = ۱۰ \times N$ است.

زیرمسئله‌ی شماره دو (۱۷ امتیاز)

- $۱ \le N \le ۱۶$ است.
- هر عدد رمزگذاری شده باید در محدوده‌ی **صفر الی** $R = ۶۵۵۳۵$ (شامل این دو عدد) باشد.
- تعداد دفعاتی که شما می‌توانید روال send را فراخوانی کنید حداکثر $K = ۱۰ \times N$ است.

زیرمسئله‌ی شماره سه (۱۸ امتیاز)

- $۱ \le N \le ۱۶$ است.
- هر عدد رمزگذاری شده باید در محدوده‌ی **صفر الی** $R = ۲۵۵$ (شامل این دو عدد) باشد.
- تعداد دفعاتی که شما می‌توانید روال send را فراخوانی کنید حداکثر $K = ۱۰ \times N$ است.

زیرمسئله‌ی شماره چهار (۲۹ امتیاز)

- $۱ \le N \le ۳۲$ است.

- هر عدد رمزگذاری شده باید در محدوده‌ی **صفر الی** $\boldsymbol{R = ۲۵۵}$ (شامل این دو عدد) باشد.
- تعداد دفعاتی که شما می‌توانید روال send را فراخوانی کنید حداکثر $\boldsymbol{K = ۱۰ \times N}$ است.

زیرمسئله‌ی شماره پنج (حداکثر ۱۹ امتیاز)

- $\boldsymbol{۱۶ \le N \le ۶۴}$ است.
- هر عدد رمزگذاری شده باید در محدوده‌ی **صفر الی** $\boldsymbol{R = ۲۵۵}$ (شامل این دو عدد) باشد.
- تعداد دفعاتی که شما می‌توانید روال send را فراخوانی کنید حداکثر $\boldsymbol{K = ۱۵ \times N}$ است.
- **نکته مهم:** امتیاز این زیرمسئله وابسته به نسبت بین طول پیام رمزگذاری شده و طول پیام اصلی است.

برای یک تست $\boldsymbol{t}$ در این زیرمسئله، $\boldsymbol{P_t = L_t/N_t}$ را نسبت بین طول پیام رمزگذاری (L_t) و طول پیام اصلی (N_t) می‌گیریم. فرض کنید $\boldsymbol{P}$ برابر با بیشینه‌ی تمام P_t ها باشد. در این صورت امتیاز شما در این زیرمسئله با قوانین زیر محاسبه می‌شود:

- اگر $P \le ۵$ باشد، شما امتیاز کامل این زیرمسئله یعنی ۱۹ امتیاز را می‌گیرید.
- اگر $۵ < P \le ۶$ باشد، شما ۱۸ امتیاز می‌گیرید.
- اگر $۶ < P \le ۷$ باشد، شما ۱۷ امتیاز می‌گیرید.
- اگر $۷ < P \le ۱۵$ باشد، امتیاز شما برابر با $۱ + ۲ \times (۱۵ - P)$ (رُند شده به پایین) خواهد بود.
- اگر $P > ۱۵$ باشد یا حداقل یکی از خروجی‌های شما نادرست باشد، شما صفر می‌گیرید.

نکته مهم: هر راه‌حل معتبری برای زیرمسئله‌های یکم الی چهارم، تمامی زیرمسئله‌های قبلی‌اش را هم حل می‌کند. اگرچه با عنایت به محدوده‌ی بزرگتر برای K در زیرمسئله‌ی پنجم، یک راه‌حل معتبر برای زیرمسئله‌ی پنجم ممکن‌ست زیرمسئله‌های یکم تا چهارم را حل نکند، ولی راه حلی هست که همه زیرمساله‌ها (یکم تا پنجم) را با هم حل می‌کند.

جزئیات پیاده‌سازی

محدودیت‌ها

- **محیط ارزشیابی:** در محیط ارزشیابی، ارسال‌های شما در دو برنامه‌ی مجزای **e** و **d** کامپایل شده و به‌صورت مجزا اجرا می‌شوند. هر دو روال رمزگذار و رمزگشا (encoder و decoder) به هر دو برنامه‌ی اجرایی لینک می‌شوند، منتهی e تنها روال encode و d تنها روال decode را صدا می‌زند.
- **محدودیت زمانی CPU:** برنامه‌ی **e** به‌تعداد ۵۰ دفعه روال encode را فرا می‌خواند و این کار باید در ۲ ثانیه انجام شود. برنامه‌ی **d** نیز ۵۰ دفعه روال decode را فرا می‌خواند و این کار نیز باید در ۲ ثانیه انجام شود.
- **محدودیت حافظه:** ۲۵۶ مگابایت
- **توجه:** هیچ محدودیت صریحی روی اندازه‌ی حافظه‌ی پشته (stack) استفاده شده وجود ندارد. میزان حافظه‌ی استفاده شده در پشته به‌عنوان بخشی از حافظه‌ی مصرفی محاسبه می‌شود.

واسط API

- فولدرِ پیاده‌سازی: /parrots

- فایل‌هایی که توسط شرکت‌کننده می‌بایست پیاده‌سازی شوند:
  - encoder.c یا encoder.cpp یا encoder.pas
  - decoder.c یا decoder.cpp یا decoder.pas

توجه برای برنامه‌نویسان به‌زبان C/C++: هر دو فایل encode.c[pp] و decode.c[pp] شما در مصحح نمونه و در مصحح اصلی با هم به مصحح لینک می‌شوند. بنابراین، شما می‌بایست تمام متغیرهای عمومی در هر فایل را به‌صورت `static` تعریف کنید تا از تداخل آن‌ها با متغیرهای دیگر فایل‌ها اجتناب شود.

- واسط‌های شرکت‌کننده:
  - encoder.h یا encoder.pas
  - decoder.h یا decoder.pas
- واسط مصحح:
  - encoderlib.h یا encoderlib.pas
  - decoderlib.h یا decoderlib.pas
- مصحّحِ نمونه: grader.c یا grader.cpp یا grader.pas

مصححِ نمونه، ۲ دور برنامه‌ی شما را اجرا می‌کند. در هر دور، ابتدا روال `encode` شما با داده‌های ورودی را اجرا کرده و سپس روی خروجی تولید شده توسط این روال، روال `decode` را اجرا می‌کند. در دور اول، مصحح ترتیب اعداد صحیح در پیام رمزگذاری شده را تغییر نمی‌دهد. منتهی در دور دوم، مصحح جای اعداد در موقعیت‌های زوج با فرد را swap می‌کند. مصحح اصلی جایگشت‌های متفاوت دیگری را روی پیام رمزگذاری شده اِعمال می‌کند. شما می توانید روش بُر زدن داده‌ها توسط مصحح نمونه را در روال `shuffle` (در C/C++) و در روال `Shuffle` (در پاسکال) تعویض کنید.

همچنین مصحح نمونه هم بر روی محدوده و هم طول داده‌های رمزگذاری شده، چک‌ای انجام می‌دهد. به‌طور پیش‌فرض مصحح نمونه بررسی می‌کند که اولاً هر عدد صحیح در داده‌های رمزگذاری شده بین صفر تا ۶۵۵۳۵ (شامل هر دو این اعداد) باشد و طول نیز حداکثر $N \times ۱۰$ باشد. شما می‌توانید این دو میزان را با تغییر ثابت `channel_range` (از ۶۵۵۳۵ به ۲۵۵ برای مثال) و `max_expansion` (از ۱۰ به ۱۵ یا ۷، برای مثال) انجام دهید.

- ورودی مصحّح نمونه: grader.in.1، grader.in.2 و ...
- **توجه:** مصحح نمونه، ورودی را به فرمت زیر می‌خواند:
  - **خط اوّل:** مقدار N
  - **خط دوم**: لیستی از N عدد که به‌ترتیب $M[0]$، $M[1]$، ... و $M[N-۱]$ هستند.
- خروجی مورد انتظار برای ورودی مصحح نمونه: grader.expect.1، grader.expect.2 و ...

برای این مسئله، هر کدام از این فایل‌ها دقیقاً باید شامل عبارت «**Correct.**» باشند.