Папагали

Яна е голяма ентусиастка по птиците. От известно време тя се занимава с опити за предаване на съобщения на големи разстояния с помощта на специално обучени интелигентни папагали.

Мечтата на Яна е да постигне предаване на съобщение **M,** съставено от **N** (не непременно различни) числа от интервала от 0 до 255, включително. Тя има **K** броя неразличими папагали, които могат да запомнят по едно цяло число от 0 до **R**, *включително*.

Отначало Яна опитваше с проста схема: пуска птиците една след друга, като на поредната птица казва поредното число от съобщението. За съжаление, папагалите обикновено достигат целта в някакъв друг ред и изпратените числа се разбъркват. По тази схема Яна може да възстанови изпратените числа, но не и правилния ред, по който са изпратени.

Вашата задача е да разработите усъвършенствана схема, в която да е възможно възстановяването на първоначалното съобщение. Трябва да напишете програма, която да изпълнява следните две операции:

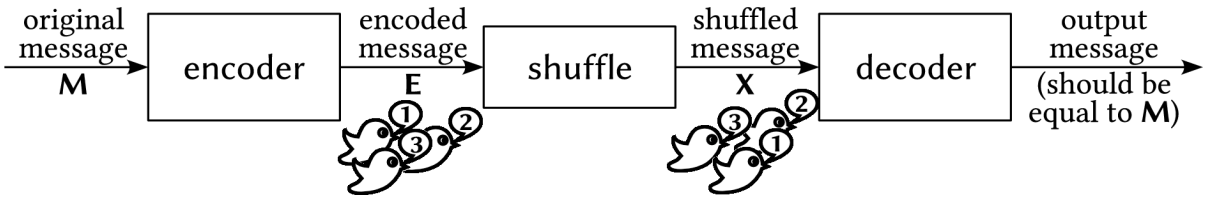
- Първо: програмата трябва да преобразува съобщение **M** в последователност от най-много **K** цели числа от интервала 0 до **R**, включително. Получените числа ще бъдат пренасяни от папагалите.
- Второ: програмата трябва, разполагайки със списъка от числата от интервала 0 до **R,** получени от птиците, да възстановява първоначалното съобщение **M**.

Предполагаме, че всички папагали достигат успешно целта и помнят числото, което им е казано. Яна напомня още веднъж, че птиците могат да пристигнат във всякакъв ред. Тя има само **K** интелигентни папагала, затова редицата от числа от интервала 0 до **R** трябва да съдържа най-много **K** числа**.**

Задача

Напишете две отделни процедури. Едната ще се използва от изпращача (кодер), а другата от получателя (декодер).

Целият процес е показан на следната фигура:



Двете процедури, които трябва да напишете са:

- Процедура **encode**(**N**,**M**), която има следните параметри:
  - **N** – дължината на съобщението.
  - **M** – едномерен масив от **N** цели числа, представляващ съобщението. Може да предполагате, че $\mathbf{0} \leq \mathbf{M[i]} \leq \mathbf{255}$ за $\mathbf{0} \leq \mathbf{i} < \mathbf{N}$.

  Тази процедура трябва да кодира съобщението **M** в последвателност от цели числа от интервала от **0** до **R**, включително, която ще бъде изпратена по папагалите. За да докладва числата от кодираното съобщение, процедурата **encode** трябва да извика

процедурата **send**(**a**) за всяко число **a**, което искате да предадете на поредната птица.

- Процедура **decode**(**N**,**L**,**X**), която има следните параметри:
  - **N** – дължината на първоначалното съобщение.
  - **L** – дължината на полученото съобщение (броят на изпратените птици).
  - **X** – едномерен масив от **L** цели числа, представляващ получените числа. Числата **X[i]** за $0 \leq i < L$ са точно тези числа, които е произвела вашата процедура **encode**, но е възможно те да бъдат разбъркани в някакъв друг ред.

  Тази процедура трябва да възстановява първоначалното съобщение. За да докладва съобщението, процедурата **decode** трябва да извика процедурата **output**(**b**) в правилния ред за всяко число **b** от декодираното съобщение.

Да отбележим, че **R** и **K** не са дадени като входни параметри – вижте описанието на подзадачите по-долу.

За правилното решаване на дадена подзадача, вашите процедури трябва да удовлетворяват следните условия:

- Всички числа изпращани от процедурата **encode** трябва да бъдат от интервала, зададен за тази подзадача.
- Броят на извикванията на процедурата **send** от процедурата **encode** не трябва да надхвърля границата **K,** зададен за подзадачата. Нека да отбележим, че **K** зависи от дължината на съобщението.
- Процедурата **decode** трябва правилно да възстановява първоначалното съобщение **M** и да извиква процедурата **output**(**b**) точно **N** пъти, с **b** равно на **M[0]**, **M[1]**, ..., **M[N–1]**.

В последната подзадача получените точки зависят от отношението на дължините на кодираното съобщение и на първоначалното съобщение.

Пример

Да разгледаме случая **N** = 3 и

M= 10
30
20

Процедурата **encode**(**N**,**M**), използвайки някакъв странен метод, може да кодира това съобщение като следната редица от числа **(7, 3, 2, 70, 15, 20, 3)**. За докладването на тази редица трябва да бъде извикана процедурата **send** както слеедва:

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

След като всички папагали са достигнали целта, да предположим, че е получена следната редица от разбъркани числа: **(3, 20, 70, 15, 2, 3, 7)**. Процедурата **decode** ще бъде извикана с

N=3, **L=7** и

3
20
70

X= 15
2
3
7

Процедурата **decode** трябва да получи първоначалното съобщение **(10, 30, 20)**. Тя докладва резултата като извиква процедурата **output** по следния начин:
output(10)
output(30)
output(20)

Подзадачи

Подзадача 1 (17 точки)

- **N = 8** и всяко число от масива **M** е или 0 или 1.
- Всяко число от кодираното съобщение трябва да бъде в интервала от **0** до **R=65535**, включително.
- Броят на извикванията на процедурата **send** трябва да бъде най-много **K=10×N**.

Подзадача 2 (17 точки)

- **1 ≤ N ≤ 16**.
- Всяко число от кодираното съобщение трябва да бъде в интервала от **0** до **R=65535**, включително.
- Броят на извикванията на процедурата **send** трябва да бъде най-много **K=10×N**.

Подзадача 3 (18 точки)

- **1 ≤ N ≤ 16**.
- Всяко число от кодираното съобщение трябва да бъде в интервала от **0** до **R=255**, включително.
- Броят на извикванията на процедурата **send** трябва да бъде най-много **K=10×N**.

Подзадача 4 (29 точки)

- **1 ≤ N ≤ 32**.
- Всяко число от кодираното съобщение трябва да бъде в интервала от **0** до **R=255**, включително.
- Броят на извикванията на процедурата **send** трябва да бъде най-много **K=10×N**.

Подзадача 5 (до 19 точки)

- $16 \leq N \leq 64$.
- Всяко число от кодираното съобщение трябва да бъде в интервала от **0** до **R=255**, включително.
- Броят на извикванията на процедурата **send** трябва да бъде най-много **K=15×N**.
- **Важно**: получените точки за тази подзадача зависят от отношението на дължините на кодираното съобщение и на първоначалното съобщение.
  Нека за отделен тест **t** в тази подзадача $P_t=L_t/N_t$ да бъде отношението на дължината L_t на кодираното съобщение към дължината N_t на първоначалното съобщение. Нека **P** да бъде най-голямото число от всички P_t. Вашите точки за тази подзадача ще се определят по следните правила:
  - Ако $P \leq 5$, ще получите пълния брой от **19** точки.
  - Ако $5 < P \leq 6$, ще получите **18** точки.
  - Ако $6 < P \leq 7$, ще получите **17** точки.
  - Ако $7 < P \leq 15$, ще получите $1 + 2 \times (15 - P)$, закръглено надолу към най-близкото цяло число.
  - Ако $P > 15$ или *някой* от вашите отговори е грешен, ще получите **0** точки.
- **Важно**: Всяко правилно решение на подзадачи 2 до 4 ще решава също и всички предишни подзадачи. Обаче, поради по-големите граници за **K**, е възможно правилно решение на подзадача 5 да не решава подзадачи от 1 до 4. Възможно да се решат всички подзадачи с едно и също решение.

Детайли по реализацията

Ограничения

- Тестваща среда: В реалната тестваща среда изпратените процедури ще бъдат компилирани в две програми **e** и **d** и ще бъдат изпълнявани отделно. И двата модула ще бъдат свързани към всяка изпълнима програма, но **e** ще вика само **encode**, а **d** – само **decode**.
- Ограничение по време: Програмата **e** ще направи 50 обръщения към процедурата **encode** и трябва да работи в рамките на 2 секунди. Програмата **d** ще направи 50 обръщения към процедурата **decode** и трябва да работи в рамките на 2 секунди.
- Ограничение по памет: 256 MB
  Бележка: Няма явна граница за размера на стековата памет. Паметта за стека се прибавя към общата използвана памет.

Интерфейс (API)

- Папка за реализация: parrots/
- Трябва да бъдат реализирани от състезателя:
  - **encoder.c** или **encoder.cpp** или **encoder.pas**
  - **decoder.c** или **decoder.cpp** или **decoder.pas**

Бележка за програмиращите на C/C++: както в примерния грейдър, така и в реалния грейдър **encoder.c**[**pp**] и **decoder.c**[**pp**] ще бъдат свързани заедно с грейдъра. **Затова, вие трябва да декларирате всички глобални променливи във всеки файл като static, за да ги предпазите от влиянието на променливи от други файлове.**

- Интерфейс на състезателя:
  - encoder.h или encoder.pas
  - decoder.h или decoder.pas
- Интерфейс на грейдъра:
  - encoderlib.h или encoderlib.pas
  - decoderlib.h или decoderlib.pas
- Примерен грейдър: grader.c или grader.cpp или grader.pas
  Примерният грейдър изпълнява два отделни кръга. Във всеки кръг грейдърът първо извиква процедурата **encode** с дадените данни, след което вика процедурата **decode** с изхода, който е произведен от **encode**. В първия крък грейдъра не променя реда на числата в кодираното съобщение. Вър втория кръг примерният грейдър размества числата на четни и нечетни позиции. Реалният грейдър ще прилага различни видове пермутации към кодираното съобщение. Може да променяте начина, по който примерния грейдър разбърква числата, като променяте неговата процедура **shuffle** (за C/C++) или Shuffle (за Pascal).
  Примерният грейдър проверява също за интервала и дължината на кодираните данни. По подразбиране се проверява дали кодираните данни са в интервала от **0** до **65535**, включително и дали дължината е най-много 10×N. Може да променяте това като променяте константите **channel_range** (от 65535 на 255, например) и **max_expansion** (от 10 на 15 или 7, например).

- Вход за примерния грейдър: grader.in.1, grader.in.2, ...
- **Бележка:** Примерният грейдър чете входа в следния формат:
  - Ред 1: N
  - Ред 2: списък от N числа: M[0], M[1], ..., M[N-1]

Очакван изход за входа на примерния грейдър : grader.expect.1, grader.expect.2, ...
За тази задача, всеки един от тези файлове трябва да съдържа единствено текста "**Correct.**".