Попугаи

Яни является любительницей птиц. С тех пор как она прочла о протоколе *IP over Avian Carriers (IPoAC)*, она провела много времени, дрессируя стаю умных попугаев для передачи сообщений на дальние расстояния.

Яни мечтает использовать своих птиц, чтобы передать сообщение **M** в очень далекую страну. Её сообщение **M** является последовательностью из **N** (не обязательно различных) целых чисел, каждое от 0 до 255 включительно. У Яни есть **K** специально натренированных попугаев. Все попугаи выглядят одинаково, Яни их не различает. Каждая птица может запоминать одно целое число от **0** до **R** *включительно*.

Вначале Яни использовала простую схему: чтобы послать сообщение, Яни аккуратно выпускала птиц из клетки одну за другой. Перед тем как каждая из птиц улетала, Яни учила птицу очередному числу из последовательности, образующей сообщение. К сожалению, эта схема не срабатывала. Оказалось, что все птицы прилетают в пункт назначения, но они не обязательно прилетают в том же порядке, в котором они улетали. Используя эту схему, Яни могла восстановить все числа, которые она отправляла, но у неё не получалось расположить их в правильном порядке.

Чтобы исполнить мечту, Яни нужна лучшая схема, и для этого Яни нуждается в вашей помощи. Имея сообщение **M**, она планирует выпускать птиц одну за другой, как и раньше. Необходимо написать программу, которая будет выполнять две отдельные операции:

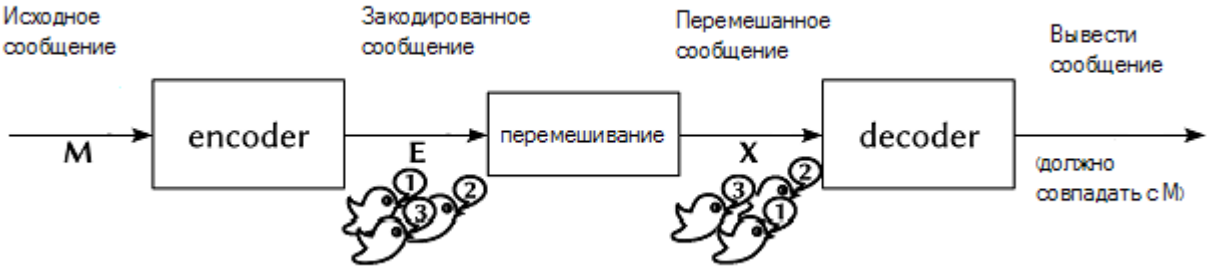
- Во-первых, программа должна читать сообщение **M** и преобразовывать его в последовательность из не более чем **K** целых чисел от **0** до **R**, которым Яни сможет обучить птиц.
- Во-вторых, программа должна читать последовательность из целых чисел от **0** до **R**, получаемых по мере достижения птицами пункта назначения, после чего преобразовать их назад в исходное сообщение **M**.

Гарантируется, что все попугаи всегда прилетают в пункт назначения и что каждый из них помнит выученное им число. Яни еще раз напоминает, что попугаи могут прилетать в произвольном порядке. Необходимо обратить внимание, что у Яни есть только **K** попугаев, то есть, последовательность целых чисел от **0** до **R**, в которую преобразуется сообщение, должна содержать не более **K** целых чисел.

Задание

Написать две отдельные процедуры. Одна из них будет использоваться при отправлении (encoder), а другая при получении (decoder).

Весь процесс показан на рисунке ниже.



Напишите две следующие процедуры:

- Процедуру encode(N,M), которой передаются следующие параметры:
  - **N** — длина сообщения.
  - **M** — одномерный массив из **N** целых чисел, которые представляют сообщение. Гарантируется, что **0** $\leq$ **M[i]** $\leq$ **255** для **0** $\leq$ **i** < **N**.

  Эта процедура должна кодировать сообщение **M** в последовательность, которая будет пересылаться с помощью попугаев и которая состоит из целых чисел от **0** до **R** включительно. Чтобы сообщить эту последовательность, процедура **encode** должна вызывать процедуру **send(a)** для каждого целого числа **a**, которому вы хотите обучить очередную птицу.
- Процедуру decode(N,L,X), которой передаются следующие параметры:
  - **N** — длина исходного сообщения.
  - **L** — длина полученного сообщения (количество отправленных птиц).
  - **X** — одномерный массив из **L** целых чисел, которые были получены. Числа **X[i]** для **0** $\leq$ **i** < **L** являются точно теми же числами, которые сгенерировала процедура encode, но они, возможно, расположены в другом порядке.

  Эта процедура должна восстановить исходное сообщение. Чтобы сообщить его, процедура **decode** должна вызывать процедуру **output(b)** для каждого целого числа **b** из расшифрованного сообщения в том порядке, в котором они образуют исходное сообщение.

Необходимо обратить внимание, что значения **R** и **K** не передаются как входные параметры (см. описание подзадач ниже).

Чтобы правильно решать определенную подзадачу, ваши процедуры должны удовлетворять следующим условиям:

- Все целые числа, отправляемые процедурой encode с помощью процедуры send, должны быть в диапазоне, указанном в подзадаче.
- Количество вызовов процедуры send, которые делает ваша процедура encode, не должно превышать предельное значение **K**, указанное в подзадаче. Необходимо обратить внимание, что значение **K** зависит от длины сообщения.
- Процедура decode должна правильно восстанавливать исходное сообщение **M** и вызывать процедуру output(b) ровно **N** раз со значениями **b**, равными числам **M[0]**, **M[1]**, ..., **M[N-1]** соответственно.

В последней подзадаче баллы, получаемые в результате оценивания, зависят от отношения между длинами закодированного и исходного сообщений.

Пример

Рассмотрим пример, где N = 3 и

M=
10
30
20

Процедура encode(N,M), используя какой-то неизвестный метод, может закодировать это сообщение следующей последовательностью чисел: **(7, 3, 2, 70, 15, 20, 3)**. Чтобы сообщить эту последовательность, она должна вызывать процедуру send в следующей последовательности:

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

Предположим, что после того как все попугаи достигли пункта назначения, был получен следующий список чисел: **(3, 20, 70, 15, 2, 3, 7)**. Процедура decode будет вызвана с **N=3**, **L=7** и

X=
3
20
70
15
2
3
7

Процедура decode должна восстановить исходное сообщение, то есть, **(10, 30, 20)**. Она сообщит результат, вызывая процедуру output в следующей последовательности:

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

Подзадачи

Подзадача 1 (17 баллов)

- **N = 8** и каждое целое число в массиве **M** равно 0 или 1.
- Каждое закодированное целое число должно быть в диапазоне от **0** до **R=65535** включительно.
- Количество вызовов процедуры send должно быть не более **K=10×N**.

Подзадача 2 (17 баллов)

- **1 ≤ N ≤ 16**.
- Каждое закодированное целое число должно быть в диапазоне от **0** до **R=65535** включительно.
- Количество вызовов процедуры send должно быть не более **K=10×N**.

Подзадача 3 (18 баллов)

- **1 ≤ N ≤ 16**.
- Каждое закодированное целое число должно быть в диапазоне от **0** до **R=255**, включительно.
- Количество вызовов процедуры send должно быть не более **K=10×N**.

Подзадача 4 (29 баллов)

- **1 ≤ N ≤ 32**.
- Каждое закодированное целое число должно быть от **0** до **R=255**, включительно.
- Количество вызовов процедуры send должно быть не более **K=10×N**.

Подзадача 5 (до 19 баллов)

- **16 ≤ N ≤ 64**.
- Каждое закодированное целое число должно быть от **0** до **R=255**, включительно.
- Количество вызовов процедуры send должно быть не более **K=15×N**.
- **Внимание**: количество баллов за эту подзадачу зависит от отношения между длинами закодированного и исходного сообщений.

  Для каждого теста с номером **t** в этой подзадаче пусть величина $\mathbf{P_t=L_t/N_t}$ будет равна отношению между длиной закодированной последовательности $\mathbf{L_t}$ и длиной исходной последовательности $\mathbf{N_t}$. Пусть **P** будет максимумом среди всех $\mathbf{P_t}$. Количество баллов, получаемых в результате оценивания за эту подзадачу, будет определяться следующими правилами:

  - Если **P ≤ 5**, решение получает за эту подзадачу все **19** баллов.
  - Если **5 < P ≤ 6**, решение получает за эту подзадачу **18** баллов.
  - Если **6 < P ≤ 7**, решение получает за эту подзадачу **17** баллов.
  - Если **7 < P ≤ 15**, количество баллов, которое решение получает за эту подзадачу, будет равно значению выражения **1 + 2 × (15 - P)**, округленному вниз до ближайшего целого числа.
  - Если **P > 15** или *хотя бы один из* ваших ответов неверен, решение получает за эту подзадачу **0** баллов.

- **Внимание**: Любое правильное решение для подзадач от 1 до 4 решает все предыдущие подзадачи. Однако из-за более высокого ограничения на **K**, правильное решение для пятой подзадачи может не решать подзадачи с 1 по 4. Есть возможность решить все подзадачи одним решением.

Детали реализации

Ограничения

- Система оценивания: В реальной системе оценивания ваше решение будет скомпилировано в две программы — **e** и **d**, которые будут запускаться отдельно. Оба модуля encoder и decoder будут прилинкованы к каждому из исполняемых файлов, но программа **e** будет вызывать только процедуру encode, а программа **d** будет вызывать только процедуру decode.
- Ограничение по времени: Программа **e** будет 50 раз вызывать процедуру encode и она должна выполняться не более 2 секунд. Программа **d** будет 50 раз вызывать процедуру decode и она должна выполняться не более 2 секунд.
- Ограничение по памяти: 256 MB
  Замечание: Нет отдельного ограничения на размер стека; используемая стеком память входит в общий объём используемой памяти.

Интерфейс (API)

- Папка для разработки: parrots/
- Участник должен разработать:
  - encoder.c или encoder.cpp или encoder.pas
  - decoder.c или decoder.cpp или decoder.pas

  Замечание при реализации на языках C/C++: как в предлагаемом модуле оценивания, так и в реальном модуле, encoder.c[pp] и decoder.c[pp] линкуются с модулем оценивания в один файл. Таким образом, чтобы избежать конфликта между переменными в модулях, необходимо все глобальные переменные в каждом из файлов объявить статическими (static).
- Интерфейс участника:
  - encoder.h или encoder.pas
  - decoder.h или decoder.pas
- Интерфейс модуля оценивания:
  - encoderlib.h или encoderlib.pas
  - decoderlib.h или decoderlib.pas
- Предлагаемый модуль оценивания: grader.c или grader.cpp или grader.pas

  Предлагаемый модуль оценивания работает в два этапа. На каждом этапе он вначале вызывает процедуру encode с необходимыми данными, после чего вызывает процедуру decode с выводом, который был получен от процедуры encode. На первом этапе модуль оценивания не изменяет порядок чисел в закодированном сообщении. На втором этапе предлагаемый модуль оценивания переставляет числа, стоящие на четных и нечетных позициях. Реальный модуль оценивания будет применять различные виды перестановок к закодированным сообщениям. Вы можете изменить способ перемешивания чисел, используемый предлагаемым модулем оценивания, изменяя процедуру shuffle (в языке C/C++) или Shuffle (в языке Pascal).

  Предлагаемый модуль оценивания также проверяет диапазон и длину закодированных данных. По умолчанию он проверяет, что закодированные данные находятся в диапазоне от **0** до **65535** включительно, и что длина не превосходит величины 10×N. Вы можете изменить это поведение, изменяя константы **channel_range** (например, с 65535 на 255) и **max_expansion** (например, с 10 на 15 или 7).
- Ввод для предлагаемого модуля оценивания: grader.in.1, grader.in.2, ...

  Замечание: Предлагаемый модуль оценивания читает входные данные в следующем формате:

  - строка 1: N
  - строка 2: последовательность из N целых чисел: M[0], M[1], ..., M[N-1]
- Ожидаемый вывод для прилагаемого модуля оценивания: grader.expect.1, grader.expect.2, ... Для этой задачи, каждый из перечисленных файлов должен содержать только текст **«Correct.»**.