Een vervelende bug ontrafelen ("Unscrambling a Messy Bug")

Ilshat is een programmeur die werkt aan efficiënte datastructuren. Op een dag heeft hij een nieuwe data structuur bedacht. De data structuur kan een set van *niet-negatieve* n-bit integers opslaan, waarbij n een macht van twee is. Dus $n = 2^b$ voor een niet-negatieve integer b.

De datastructuur is in het begin leeg. Een programma dat de datastructuur gebruikt, moet zich aan de volgende regels houden:

- Het programma kan n-bit integers in de datastructuur toevoegen. Dit gebeurt een voor een door de functie `add_element(x)` te gebruiken. Als het programma een element probeert toe te voegen dat al bestaat dan gebeurt er niets.
- ALs het laatste element is toegevoegd, roept het programma de functie `compile_set()` precies een keer aan.
- Tenslotte kan het programma `check_element(x)` aanroepen om te controleren of het element x in de datastructuur is opgeslagen. Deze functie mag meerdere keren aangeroepen worden.

Bij het implementeren van `compile_set()` heeft Ilshat een fout gemaakt. Deze bug zorgt ervoor dat de bits van elk element in de set op dezelfde manier hergerangschikt worden. Ilshat wil graag dat jij de precieze herschikking bepaalt die door de bug wordt veroorzaakt.

Formeel, bekijk een rij $p = [p_0, \ldots, p_{n-1}]$ waar elk getal van 0 tot en met $n-1$ precies een keer in voorkomt. Deze rij noemen we een *permutatie*. Gegeven een element in de set waarbij de bits $a_0, \ldots, a_{n-1}$ zijn (met a_0 als meest significante bit). Wanneer de functie `compile_set()` wordt aangeroepen dan wordt dit element vervangen door element $a_{p_0}, a_{p_1}, \ldots, a_{p_{n-1}}$.

Dezelfde permutatie p wordt gebruikt om de digits van elk element te herordenen. Elke permutatie is mogelijk, er is ook de mogelijkheid dat $p_i = i$ voor elke $0 \leq i \leq n-1$.

Bijvoorbeeld, stel dat $n = 4$, $p = [2, 1, 3, 0]$, en je hebt de integers opgeslagen waarvan de binaire representaties `0000`, `1100` en `0111` zijn. De aanroep van `compile_set` verandert deze elementen naar `0000`, `0101` en `1110` respectievelijk.

Schrijf een programma dat de permutatie p bepaalt door met de datastructuur te communiceren. Je programma moet (in de volgende volgorde):

1. een set van n-bit integers kiezen,

2. deze integers aan de datastructuur toevoegen,
3. de functie `compile_set` aanroepen om de bug te triggeren,
4. nagaan of bepaalde elementen in de gewijzigde set aanwezig zijn,
5. die informatie gebruiken om de permutatie p te bepalen en terug te geven.

Merk op dat je programma de functie `compile_set` maar een keer mag aanroepen.

Er zijn grenzen aan hoe vaak je programma de library functies mag aanroepen: je mag

- `add_element` maximaal w keer aanroepen (w staat voor "writes"),
- `check_element` maximaal r keer aanroepen (r staat voor "reads").

Implementatie details

Implementeer de volgende functie (methode):

- `int[] restore_permutation(int n, int w, int r)`
  - `n`: het aantal bits in de binaire representatie van elk element in de set (en ook de lengte van p).
  - `w`: het maximale aantal aanroepen van `add_element` dat je programma mag doen.
  - `r`: het maximale aantal aanroepen van `check_element` dat je programma mag doen.
  - je programma moet de herleide permutatie p als resultaat terug geven.

In de taal C is prototype van de functie een beejte anders:

- `void restore_permutation(int n, int w, int r, int* result)`
  - `n`, `w` en `r` hebben dezelfde betekenis als hierboven.
  - de functie moet de herleide permutatie p terug geven door deze op te slaan in de array `result`: voor elke i, moet je de waarde p_i opslaan in `result[i]`.

Library functies

Om met de datastructuur te communiceren moet je programma de volgende drie functies (methoden) gebruiken:

- `void add_element(string x)`
  Deze functie voegt het element `x` toe aan de set.
  - `x`: een string van `'0'` en `'1'` die de binaire representatie van een toe te voegen integer aangeeft. De lengte van `x` moet n zijn.

- `void compile_set()`
  Deze functie moet precies een keer aangeroepen worden. Je programma mag geen `add_element()` aanroepen na deze aanroep. Je programma mag geen `check_element()` aanroepen voor deze aanroep.
- `boolean check_element(string x)`
  Deze functie bepaalt of element `x` aanwezig is in de aangepaste set.
  - `x`: een string van `'0'` en `'1'` die de binarire representatie van een te controleren integer aangeeft. De lengte van `x` moet n zijn.
  - geeft `true` terug als element `x` in de aangepaste set zit en anders `false`.

Merk op dat als je programma zich niet aan de bovenstaande beperkingen houdt het resultaat van de grader "Wrong Answer" is.

Voor alle strings geldt dat het eerste character de meest significante bit van de betreffende integer aanduidt.

De grader legt de permutatie vast voordat de aanroep van `restore_permutation` wordt gedaan.

Gelieve de voorziene template bestanden te gebruiken voor implementatiedetails in jouw programmeertaal.

Voorbeeld

De grader doet de volgende functie aanroep:

- `restore_permutation(4, 16, 16)`. Gegeven $n = 4$, en je programma mag maximaal 16 "writes" en 16 "reads" doen.

Je programma doet nu de volgende functie aanroepen:

- `add_element("0001")`
- `add_element("0011")`
- `add_element("0100")`
- `compile_set()`
- `check_element("0001")` returns `false`
- `check_element("0010")` returns `true`
- `check_element("0100")` returns `true`
- `check_element("1000")` returns `false`
- `check_element("0011")` returns `false`
- `check_element("0101")` returns `false`
- `check_element("1001")` returns `false`
- `check_element("0110")` returns `false`
- `check_element("1010")` returns `true`
- `check_element("1100")` returns `false`

Er is maar een permutatie consistent met de waarden die `check_element()` opleverde: de permutatie $p = [2, 1, 3, 0]$. Dus de `restore_permutation` moet `[2, 1, 3, 0]` terug geven.

Subtaken

1. (20 punten) $n = 8$, $w = 256$, $r = 256$, $p_i \neq i$ voor hoogstens 2 indices i ($0 \leq i \leq n - 1$),
2. (18 punten) $n = 32$, $w = 320$, $r = 1024$,
3. (11 punten) $n = 32$, $w = 1024$, $r = 320$,
4. (21 punten) $n = 128$, $w = 1792$, $r = 1792$,
5. (30 punten) $n = 128$, $w = 896$, $r = 896$.

Voorbeeldgrader

De voorbeeldgrader leest de invoer in het volgende formaat:

- regel 1: integers n, w, r,

- regel 2: n integers die p beschrijven.