G’alati bagni tadqiq qilish.

Ilshod dasturchi bo’lib ishlaydi va yuqori darajali ma’lumotlar strukturasini yaratish ustida ish olib boradi. Bir kuni u yangi ma’lumotlar strukturasini yaratdi. Bu ma’lumotlar strukturasi n-bitli butun nomanfiy sonlar to’plamini o’zida saqlaydi, bunda n- ikkining darajasi. Ya’ni, qandaydir butun nomanfiy b soni uchun n=2b. Boshida ma’lumotlar strukturasi bo’sh. Bu ma’lumotlar strukturasidan foydalanuvchi dastur quyidagi qoidalarga rioya qilish kerak:

- Dastur ma’lumotlar strukturasiga n bitli butun sonlar bo’lgan elementlarni qo’shishi mumkin, bunda dastur add_ element(x) ushbu funksiyani ishlatishi kerak. Agar dastur berilganlar strukturasiga bu strukturada mavjud sonni qo’shmoqchi bo’lsa, hech nima o’zgarmaydi.
- Elementlar qo’shilgach, dastur compile_ set() funksiyasini faqat bir marta chaqirishi kerak.
- Shundan so’ng dastur check_ element(x) funksiyasini chaqirishi mumkin, bu funksiya x elementi ma’lumotlar strukturasida bor-yo’qligini tekshiradi. Bu funksiyani bir necha marta chaqirish mumkin.

Ilshod birinchi marta bu ma’lumotlar strukturasini yozganida, u compile *set() bag(xato) qilib qo’ydi. Natijada bu funksiya chaqirilgach, har bir elementning ikkilik sanoq sistemasidagi raqamlari joylashuvi o’zgarib ketadi (har bir elementni yaqinlari o’z ichida joylashadi), bunday barcha elementlarning raqa,larining o’rin almashinishi bir xil qonuniyatga ko’ra ro’y berar ekan. Ilshod raqamlar aynan qay tartibda o’rin almashinayotganini aniqlamoqchi.*
P0,…,Pn-1 ketma-ketligini ko’rib chiqaylik. Bu ketma-ketlikda 0 dan n-1 gacha bo’lgan har bir son bir martadan uchraydi. Bunday ketma-ketlikni perestanovka deb ataymiz. Ma’lumotlar strukturasini ikkilik sanoq sistemasidagi yozuvi a0,…an-1 raqamlaridan iborat elementni ko’rib chiqamiz (bunda a0 – yuqori bit ya’ni, sonning eng chapidagi raqam). compile set() funksiyasi chaqiriliganda, bu element ikkilik yozuvi ap0, ap1, …,apn-1(izoh:0,1,n-1 – p ning indeksi) bo’lgan elementga almashtiriladi.
p perestanovkasi har bir elementning raqamlari tartibni o’zgartirish uchun ishlatiladi. Perestanovka ixtiyoriy bo’lishi mumkin, xususan, ayniy ya’ni, pi=i bo’lishi mumkin (0<=i<=n-1)
Masalan, n-4, p[2,1,3,0], va dastur berilganlar strykturasiga ikkilik ko’rinishi 0000, 1100 va 0111 bo’lgan elementlarni qo’shgan bo’lsin. compile_ set funksiyasini chaqirish bu elementlarni 0000, 0101 va 1110 elementlariga mos ravishda almashtiradi.
Ma’lumotlar strukturasi bilan jarayonga kirishib, p perestanovkani topuvchi dastur yoishingiz kerak. Bu dastur birin-ketin ushbu amallarni bajarsin:

1. n bitli butun nomanfiy sonlar to’plamini tanlash;
2. bu elementlarni ma’lumotlar strukturasiga qo’shish;

3. bagni faollashtirish uchun compile_ set funksiyasini chaqirish;
4. hosil bo’lgan ma’lumotlar strukturasidan ba’zi bir elementlar bor-yo’qligini tekshirish;
5. olingaan ma’lumotlarni ishlatib, p perestanovkasini aniqlash va qaytarish.

E’tibor qiling: compile_ set funksiyasini faqat 1 marta chaqirish mumkin.
Shuningdek, sizni datsuringiz ma’lumotlar strukturasiga jo’natuvchi so’rovlar soni cheklangan:

- add_element funksiyasini w tadan ortiq chaqirib bo’lmaydi.(w inglizcha write so’zidan olingan)
- check_element funksiyasini r martadan ortiq chaqirib bo’lmaydi (r inglizcha read so’zidan)

Realizatsitya tafsilotlari:

Bitta funksiya yozishingiz kerak:

- int[ ] restore_permutation (int n, int w, int r)
- n: berilganlar strukturasidagi har bir elementning ikkilik yozuvidagi bitlar soni (shunigdek, qidirilayotgan p perestanovka uzunligi)
- w: add_element funksiyasini maksimal chaqiruvi soni.
- r: check_element funksiyasini maksimal chaqiruvi soni.
- Funksiya topilgan p perestanovkani qaytarishi kerak

C tilida funksiya ramzlari biroz farqlanadi:

void restore_permutation (int n, int w, int r, int* result)

- n,w,va r yuqoridagidek.
- Funksiya topilgan p perestanovkani result massiviga joylashtirib, qaytarishi kerak: har bir i uchun pi qiymatini result[i] ga joylashtirishi kerak.

Kutubxona funksiyalari:

Ma’lumotlar strukturasi bilan jarayonga kirishish uchun dasturingiz ushbu 3 funksiyani ishtashi kerak:

- void add_element(string x)
  x qatori bila berilgan elementni berilganlar strukturasiga qo’shish.
- x: ma’lumotlar strukturasiga qo’shish kerak bo’lgan sonning ikkilik ko’rinishini beruvchi 0 va 1 dan iborat qator. x qatori uzuznligi n gat eng bo’lishi kerak.
- void compile_set()
  Bu funksiya bir marta chaqirilishi kerak. Dasturingiz add_element() funksiyasini bu chaqiruvdan so’ng chaqirolmaydi. Bu chaqiruvgacha check_element() ham chaqirilmaydi.
- boolean check_element(string x)
  Bu funksiya compile_set() chaqiruvidan so’ng hosil bo’lgan ma’lumotlar strukturaisda x qatori bilan berilgan element borligini tekshiradi.
- x : ma’lumotlar strukturasida borligini tekshirish kerak bo’lgan sonning ikkilik ko’rinishini beruvchi 0 va 1 dan iborat qator. x qatori uzunligi n ga teng bo’lishi kerak.
- x ma’lumotlar strukturasida bor bo’lsa, true, yo’q bo’lsa false
  Agar dastur yuqoridagi cheklovlarni birini buzsa, natija wrong answer bo’ladi.

Example

The grader makes the following function call:

- `restore_permutation(4, 16, 16)`. We have $n = 4$ and the program can do at most 16 "writes" and 16 "reads".

The program makes the following function calls:

- `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`

Only one permutation is consistent with these values returned by `check_element()`: the permutation $p = [2, 1, 3, 0]$. Thus, `restore_permutation` should return `[2, 1, 3, 0]`.

Subtasks

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

Sample grader

The sample grader reads the input in the following format:

- line 1: integers n, w, r,
- line 2: n integers giving the elements of p.