Tradisjonsmat

På Vestlandet er det alltid mange turister som har lyst til å prøve merkelige norske tradisjonsretter, gjerne etter å ha besøkt en lokal foss. Du har nylig fått jobb i en kiosk som selger boller med brunost. Din jobb er å sette sammen to bolle-halvdeler med en skive brunost i midten. Kollegaen din med ansvar for å kutte opp bollene er veldig vimsete. Du får derfor hver morgen et lass av halve boller, av **ujevn tykkelse**, som du skal bruke. Perfeksjonist som du er, vil du at alle bollene du selger skal være **like store**.

Jobben din er å smøre N like store boller. For å hjelpe, bestemmer du deg for å lage et program som gitt en liste av vekten til $2N$ halve boller, forteller deg hvilke halve boller som skal settes sammen for at du skal kunne lage N boller med lik vekt.

Input

Første linje består av heltallet N. Så følger en linje med $2N$ heltall, en for hver halve bolle, nummerert fra 0 til $2N-1$. Det i-te tallet beskriver da vekten w_i til bollehalvdel i.

Output

Skriv ut N linjer. Hver linje skal bestå av to heltall, i og j, de to bollehalvdelene som skal settes sammen for å lage en bolle. Alle gyldige kombinasjoner av bollehalvdelene godkjennes.

Skriv ut -1 hvis det er umulig å sette sammen bollehalvdelene slik at alle bollene får lik vekt.

Begrensninger

$1 \leq N \leq 200\ 000$

$1 \leq w_i \leq 100\ 000\ 000$ for $0 \leq i < 2N$

Tidsbegrensning: 2 s

Testsettgruppe	**Poeng**	**Ytterligere begrensninger**
Gruppe 1	7	Det finnes nøyaktig to ulike verdier for w_i. For eksmpel vekter $\{1, 2\}$ og $\{2, 3\}$.
Gruppe 2	15	Det finnes nøyaktig tre ulike verdier for w_i
Gruppe 3	7	$N = 2$
Gruppe 4	21	$N = 1000$
Gruppe 5	50	Ingen andre begrensninger

Eksempler

Input	Output	Kommentarer
2 4 6 3 1	3 1 2 0	Bollehalvdel 3 og 1 settes sammen for å lage en bolle med vekt 7. Samme for bollehalvdel 2 og 0.

Input	Output	Kommentarer
2 1 5 3 8	-1	Det er ikke mulig å lage en bolle med lik vekt av bollehalvdelene.

Løsning for testsettgruppe 1

Denne oppgaven kommer med en komplett løsning for testsettgruppe 1 slik at det skal være lettere å komme i gang med runde 2 og bli kjent med konkurransesystemet.

I testsettgruppe 1 er begrensningen at det finnes nøyaktig to ulike vekter. Siden vi har nøyaktig to ulike vekter, må vi bare sjekke om det er like mange av hver vekt. Hvis det er det, kan vi bare pare sammen de to ulike vektene.

Det kan gjøres i Python slik som dette:

```python
# Leser inn en linje med ett tall
n = int(input())

# Leser inn alle tall på en linje og lagrer som en liste
w = [int(x) for x in input().split()]

weight_light = min(w)
weight_heavy = max(w)

light_halves = []
heavy_halves = []

# enumerate returnerer hvert element i listen (weight)
# sammen med indeks (i)
for i, weight in enumerate(w):
    if weight == weight_light:
        light_halves.append(i)
    elif weight == weight_heavy:
        heavy_halves.append(i)

if len(light_halves) != len(heavy_halves):
    print(-1)
else:
    # zip av to lister gir et element fra hver av
    # listene for hver iterasjon.
    # Altså er i og j elementer med samme
    # indeks i light_halves og heavy_halves
    for i, j in zip(light_halves, heavy_halves):
        # Skriver ut to tall, separert med mellomrom
        print(i, j)
```

Tannhelse

Du har fått i oppdrag å hjelpe arkeologer med å studere tannhelsen til gamle fossiler. Et fossil har en rekke med N tenner, nummerert fra 0 til $N-1$. Om tann nr. i har vært infisert så vil det ha ført til skade på minst én, men potensielt flere, av tennene $\max(0, i-1)$, tann i og tann $\min(n-1, i+1)$, altså den infiserte tannen eller de to tilstøtende. Alle tannskader kommer fra slike infeksjoner.

Gitt en oversikt over skadde tenner på et fossil, finn det minste og største potensielle infeksjoner i tannrekken.

Input

Første linje i input består av heltallet N.

Så følger en linje med N heltall, nummerert fra 0 til $N-1$. Om tall nummer i er 0 så er tann nr. i ikke skadet. Om tall nummer i derimot er 1 så er tann nummer i skadet.

Output

Skriv ut to heltall, minste og største antall tenner som potensielt sett kan ha vært infisert.

I hver testsettgruppe får du halvparten av mulige poeng om du alltid skriver ut riktig nedre grense, og halvparten av poengene om du alltid skriver ut korrekt øvre grense. Merk at du alltid likevel må skrive ut to tall, så om programmet ditt for eksempel beregner den øvre grensen må det likevel skrive ut ett tall (for eksempel 0) som den nedre grensen først, og så tallet for den øvre grensen.

Begrensninger

$1 \leq N \leq 10^6$

Tidsbegrensning: 2 s

Testsettgruppe	**Poeng**	**Ytterligere begrensninger**
Gruppe 1	8	$N = 1$
Gruppe 2	10	$N = 3$
Gruppe 3	10	Det er minst to friske tenner mellom hver syke tann.
Gruppe 4	12	Alle tennene er skadet.
Gruppe 5	60	Ingen andre begrensninger

Merk igjen at i hver testsettgruppe, også kalt deloppgave, får du halvparten av mulige poeng om du alltid skriver ut riktig nedre grense, og halvparten av poengene om du alltid skriver ut korrekt øvre grense. For å få full pott i en deloppgave må altså nedre og øvre grense alltid være riktig.

Eksempler

Input	Output	Kommentarer
4 1 0 0 0	1 2	Minst en av tennene må være infisert, siden vi har en tannskade, og alle tannskader kommer fra infeksjoner. Det er derimot mulig at både tann 0 og tann 1 har vært infisert. Det er ikke mulig at for eksempel tann 2 har vært infisert, siden ingen av nabotannene eller tannen selv har en skade.

Klatrevegger

Birk har lenge jobbet med klatrevegger. En av kjerneoppgavene til Birk er å avgjøre vanskelighetsgraden til en klatrevegg. Vanskelighetsgraden til en klatrevegg er det laveste ferdighetsnivået en klatrer må ha for å kunne utforske ethvert punkt i veggen, fra ethvert punkt som utgangspunkt, uten hjelp. Hvis det ikke er mulig for en klatrer selv med ubegrensede ferdigheter å utforske veggen slik, sier vi at klatreveggen er umulig.

En klatrevegg beskrives som en mengde punkter og potensielle hopp mellom punkter. Disse hoppene krever et gitt minimum av ferdighet for å kunne gjennomføre. Alle slike ferdighetsnivåer er gitt av heltall.

Gitt en liste over punktene på klatreveggen og alle potensielle hopp sammen med deres krav til ferdighet, finn vanskelighetsgraden til klatreveggen eller konkluder at klatreveggen er umulig.

Input

Første linje består av to heltall, N og K, antall punkter og antall potensielle hopp eller koblinger mellom punktene. Vi nummererer punktene fra 0 til $N-1$.

Så følger K linjer, hvor hver linje beskriver en potensiell kobling. Hver av disse linjene inneholder tre heltall a, b, $f_{a,b}$, hvor a, b er de to punktene som kobles sammen og $f_{a,b}$ er minimum ferdighetsnivået som kreves for å bevege seg fra a til b og fra b til a. Vi garanterer at $f_{a,b}$ er et heltall større enn eller lik 0, og at det finnes maksimalt én direkte kobling mellom hver a, b.

Output

Output skal bestå av ett heltall. Om klatreveggen er umulig skriver du ut -1. Ellers, skriv ut vanskelighetsgraden til klatreveggen, altså det minste tallet v slik at det for en klatrer med ferdighetsnivå minst v er mulig å bevege seg mellom ethvert par av punkter i klatreveggen.

Begrensninger

$1 \leq N \leq 100\ 000$

$1 \leq K \leq 100\ 000$

$0 \leq f_{a,b} \leq 100\ 000\ 000$ for alle a, b hvor det finnes en kobling.

Tidsbegrensning: 2 s

Testsettgruppe	Poeng	Ytterligere begrensninger
Gruppe 1	3	$N = 2$
Gruppe 2	5	$N = 3$
Gruppe 3	10	For hvert par av punkter finnes nøyaktig én sekvens med hopp for å komme seg fra det ene punktet til det andre
Gruppe 4	19	$N \leq 40$ og $f_{a,b} \leq 40$
Gruppe 5	23	$N \leq 40$
Gruppe 6	40	Ingen andre begrensninger

Eksempler

Input	Output	Kommentarer
6 7 0 1 4 1 2 5 4 5 1 1 5 7 0 2 8 2 4 0 4 3 2	5	Med ferdighetsnivå lik 5 er det mulig å bevege seg mellom ethvert par av punkter. Med et lavere ferdighetsnivå vil det for eksempel ikke være mulig å bevege seg mellom punkt 1 og punkt 3.

Input	Output
4 2 0 1 4 1 3 2	-1

Romfart

Som en del av sikring av en fremtid for mennesker etter jordens uunngåelige undergang, skal N satellitter sendes mot en måne ved navn Fremtid. Fremtid og jorden ligger et ukjent antall lyssekunder unna hverandre. Å reise ett lyssekund koster én enhet med drivstoff.

Fysikere har nylig funnet ut at avstanden mellom jorden og Fremtid er et heltall antall lyssekunder x slik at $A \leq x \leq B$. I tillegg vet de at sannsynligheten for at $x = i$ for i mellom A og B er konstant, altså kan vi si at avstanden fra jorden til Fremtid kan modelleres som et uniformt tilfeldig trukket heltall mellom A og B.

Menneskeheten ønsker å bruke minst mulig drivstoff på å sende N satellitter mot Fremtid. Om Fremtid er x lyssekunder unna jorden vil en satellitt utrustet med minst x enheter drivstoff trygt ankomme Fremtid, mens en satellitt med mindre enn x enheter drivstoff vil forsvinne på vei til Fremtid. Enten ved at satelitten lander eller går tom for drivstoff, vil det sendes ut et signal om dette tilbake til jorden. Det er et ubegrenset antall satellitter på jorden, men drivstoff er en begrenset ressurs. Du har fått i oppdrag å lage en algoritme som bestemmer hvor mye drivstoff som skal sendes på hver av satellittene, én etter én, helt til N av dem har rapportert at de er landet på Fremtid.

Fysikerne på teamet vil teste løsningen din mot flere hypotetiske verdier av A, B og N, før de eventuelt bestemmer om de skal ta i bruk løsningen din i den ekte verden.

Input

Første linje i input består av tallet R, som bestemmer hvor mange runder programmet ditt vil kjøres.

Hver runde starter med en linje input, som består av tallene A, B og N. Input etter dette avhenger av interaksjon.

Interaksjon

For å sende en rakett med k enheter drivstoff, skriv ut `send k`. Avhenging av om satellitten lander eller ikke vil det skrives enten `landet` eller `forsvunnet`.

En runde avsluttes etter at `landet` er skrevet ut fra graderen N ganger.

Merk at stdout blir bufret, det kan derfor være nødvendig å flushe bufferen for hver skriving for å unngå forsinkelser i interaksjonen. I C++ bør du avslutte linjer du printer med `std::endl`. I Python kan du legge til `flush=True` i `print`, eller kalle `sys.stdout.flush()`. I Java flushes stdout automatisk hvis du bruker `System.out.println()`.

Begrensninger

$1 \leq A \leq B \leq 1000$

$1 \leq N \leq 1000$

$1 \leq R \leq 100$

Tidsbegrensning: 2 s

Scoring

I deloppgave 1, 2 og 3 får du poeng etter hvor nær løsningen din er NIO sin løsning. Hvis løsningen din bruker mindre eller like mye drivstoff som NIO sin løsning, så vil du få full uttelling. Deloppgavene består av flere testcaser. Scoren din i en gitt deloppgave er lik summen av scoren for hver testcase, så lenge løsningen din i alle tilfeller gir gyldige svar. Om løsningen din i en test-case i snitt bruker x prosent mer drivstoff en NIO sin løsning, får du $T \cdot 2^{-x}$ poeng i den testcasen, hvor T er maksimalt oppnåelig antall poeng for den spesifike testcasen. Dette betyr for eksempel at om du bruker 1% mer drivstoff enn NIO sin løsning, får du halvparten av full uttelling i testcasen. Husk at hver testcase består av flere runder, så summen av drivstoffbruk for hver test-case består av summen over alle disse rundene.

I deloppgave 4 får du poeng etter hvor nær du er den teoretisk optimale løsningen som innebærer å korrekt tippe avstand mellom jorden og Fremtid. Om du i snitt i en test-case bruker x prosent mer drivstoff enn løsningen som inebærer å korrekt tippe avstanden, får du $T \cdot 2^{-x/10}$ poeng, hvor T er totalt antall poeng for den test-casen. Dette betyr for eksempel at om du bruker 10% mer drivstoff, får du halvparten av full uttelling i testcasen.

Testsettgruppe	**Poeng**	**Ytterligere begrensninger**
Gruppe 1	4	$N = 1$
Gruppe 2	7	$A = 1, B = 2$
Gruppe 3	55	$N \leq 15, B \leq 50$
Gruppe 4	34	$N \geq 30$

Eksempel

Linjer med $>$ foran er output fra programmet ditt og linjer med $<$ foran er `input` som programmet ditt får.

```
< 2
< 12 15 2
> send 12
< forsvunnet
> send 14
< landet
```

```
> send 13
< landet
< 18 18 1
> send 18
< landet
```

Logistikkdrøm

Du venter på den innsjekkede bagasjen din på hjemreisen fra en NIO-finale, og faller til dagdrømming om hvordan du kanskje kunne hjulpet til med å effektivisere bagasjetransporten internt på flyplassen. Siden du ikke aner hvordan logistikken faktisk ser ut, ser du heller for deg følgende design:

All bagasje på flyplassen skal fraktes på et kvadratisk rutenett, med størrelse $N \times N$, fra ruten nederst til venstre som har koordinater $(1, 1)$ til ruten øverst til høyre som vi gir koordinater (N, N). I hver rute er det i utgangspunktet enten ingenting, eller et bånd som dytter kofferter én rute opp, ned, til høyre eller til venstre etter ett sekund. Mer formelt, om det i koordinat (x, y) er et bånd, så vil en koffert som er i koordinat (x, y) på tid S, kofferten være i koordinat $(x + a, y + b)$ på tid $S + 1$, der

- $a = 0, b = 1$ hvis båndet dytter opp
- $a = 0, b = -1$ hvis båndet dytter ned
- $a = 1, b = 0$ hvis båndet dytter til høyre
- $a = -1, b = 0$ hvis båndet dytter til venstre

Om det ikke er et bånd i koordinat (x, y), vil en koffert som står i koordinat (x, y) i tid S og så stå i samme koordinat i tid $S + 1$.

I tillegg er det ingen bånd i randsonen, altså i området med koordinater (x, y) der $x = 1$ eller $y = 1$, eller $x = N$ eller $y = N$.

Gitt at en koffert er plassert i koordinat $(1, 1)$ i tid $S = 0$, finn et minste antall ruter som må endres (ved å legge til bånd, fjerne bånd eller endre bånd) slik at kofferten ankommer koordinat (N, N) etter minst mulig tid.

Input

Input består først av heltallet N og K, hvor K er antall bånd utplassert. Så følger K linjer på formen "x y t" hvor x, y er heltall mellom 2 og $N - 2$ og t er en av bokstavene "o","n","h", eller "v", som beskriver båndet av type "opp", "ned", "høyre" eller "venstre" respektivt.

Output

Output består av ett heltall som er antallet ruter som må endres.

Begrensninger

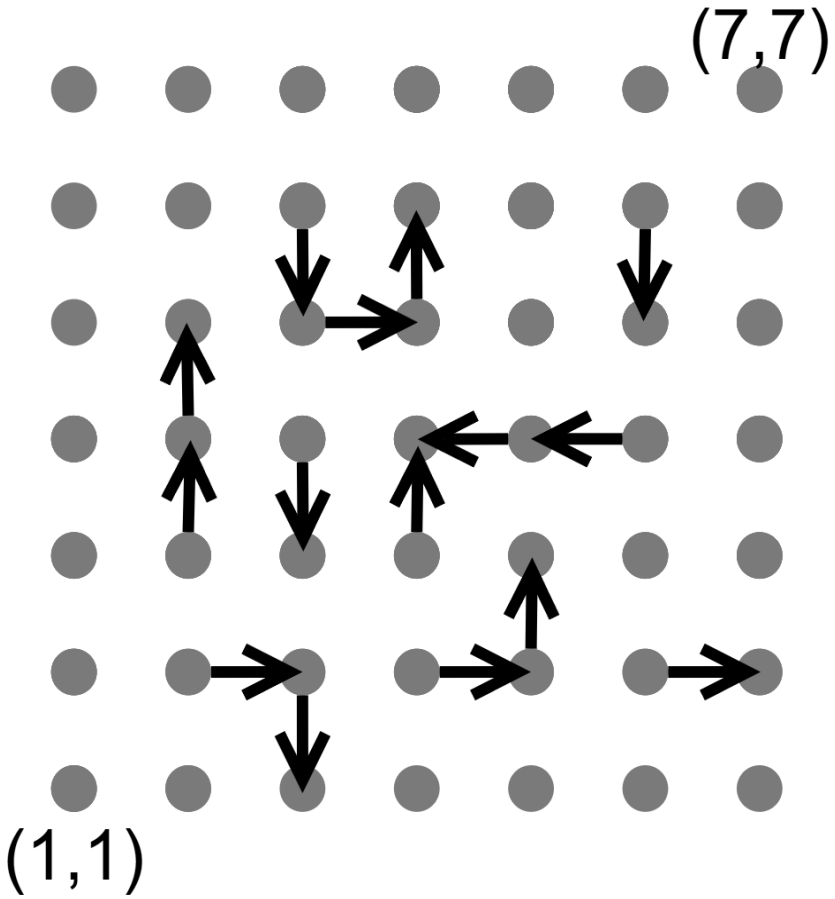
$3 \leq N \leq 20\ 000$

$1 \leq K \leq min((N-2)^2, 100\ 000)$

Testsettgruppe	**Poeng**	**Ytterligere begrensninger**
Gruppe 1	3	$K = 1$
Gruppe 2	5	$K = 2$
Gruppe 3	10	$K \leq 1\ 000$
Gruppe 4	23	$N \leq 1\ 000$
Gruppe 5	27	Alle bånd utplassert peker i utgangspunktet oppover, altså er av type "o"
Gruppe 6	32	Ingen andre begrensninger

Eksempel 1

Input	**Output**
7 15 2 2 h 3 2 n 4 2 h 5 2 o 6 2 h 2 3 o 4 3 o 2 4 o 3 4 n 5 4 v 6 4 v 3 5 h 3 6 n 4 5 o 6 6 n	8



Pilene representerer bånd som starter i en rute og som frakter til ruten pilen peker på.