Садржај

1 1. круг квалификација 3
- Задатак: Године . . . 3
- Задатак: Чоколада . . . 4
- Задатак: Јмбг . . . 5
- Задатак: Комбинација задатака . . . 7
- Задатак: Број сегмената парног збира . . . 8
- Задатак: Контролна цифра . . . 11
- Задатак: Најубедљивија победа . . . 12
- Задатак: Јединичне колоне . . . 13
- Задатак: Постојање троугла . . . 14
- Задатак: Аутопревозник . . . 16
- Задатак: Термометар . . . 17
- Задатак: Једначина . . . 18
- Задатак: Највећа комбинација . . . 19

2 2. круг квалификација 21
- Задатак: Гринготс . . . 21
- Задатак: Путовање . . . 22
- Задатак: Приближан рачун . . . 22
- Задатак: Добри парови . . . 24
- Задатак: Цензура . . . 24
- Задатак: Ширење тајне . . . 25
- Задатак: Хари Потер . . . 28
- Задатак: Квазинаучник . . . 29
- Задатак: Да ли постоји правоугаоник . . . 31
- Задатак: Сума свих поднизова . . . 32
- Задатак: Слагалица . . . 33
- Задатак: Гумене бомбоне . . . 34
- Задатак: Адвокатица . . . 35
- Задатак: Слатки поднизови . . . 36

3 Општинско такмичење 39
- Задатак: Буџет за летовање . . . 39
- Задатак: Упоредити цифре . . . 39
- Задатак: Дељивост са 3 7 21 . . . 41
- Задатак: Распоређивање слика . . . 41

Задатак: Свеска 42
Задатак: Верзије софтвера 43
Задатак: Жуђков реjон 45
Задатак: Јелка 46
Задатак: Питагорина тројка 46
Задатак: Учешће 47
Задатак: Баундинг бокс 49
Задатак: Различите мајице 49

4 Окружно такмичење 51
Задатак: Нз 51
Задатак: Другарице 52
Задатак: Пикадо 53
Задатак: Ски стаза 55
Задатак: Седласти елемент 58
Задатак: Наводњавање 60
Задатак: Заборавна плесачица 61
Задатак: Већи од свих парних 62
Задатак: Бриц 64

Глава 1

1. круг квалификација

Задатак: Године

Аутори: Нина Икодиновић, Михајло Марковић

Данас је Сара нашла фотографију са Y -тог рођендана другарице Мање, која је настала истог датума као и данас. Тада је Сара имала X година. Ако данас Сара има Z година, помозите јој да одреди колико година Мања пуни данас како би могла да је позове и честита јој рођендан.

Опис улаза

У првом реду уноси се природан број X ($1 \leq X \leq 100$), број Сариних година на фотографији.

У другом реду уноси се природан број Y ($1 \leq Y \leq 100$), број Мањиних година на фотографији.

У трећем реду уноси се природан број Z ($X < Z \leq 100$), број Сариних година данас.

Опис излаза

У једном реду стандардног излаза исписати један број који представља колико Мања данас пуни година.

Пример 1

Улаз	*Излаз*	*Објашњење*

`10`
`14`
`36`
	`40`	Када је Сара имала 10 година, Мања је пунила 14. Данас Сара има 36 година, па Мања пуни 40.

Пример 2

Улаз	*Излаз*

`17`
`13`
`66`
	`62`

Решење

Од тренутка настанка фотографије до данас прошло је $Z - X$ година. Према томе, Мања данас пуни $Y + (Z - X)$ година.

```
x=int(input())
y=int(input())
z=int(input())
g=y+z-x
print(g)
```

Задатак: Чоколада

Аутор: Теодора Обрадовић

Неца и Цеца су брат и сестра који се стално свађају. Њихова богата тетка из Немачке није знала за то и купила им је једну огромну чоколаду. С обзиром да обоје желе највеће парче чоколаде, договорили су се да Неца подели чоколаду на два дела, а да Цеца бира који део жели. Неца је знао да ће Цеца хтети да одабере већи део, па је он на брзину узео и сакрио један део, а њој показао други (за који она не зна да ли је већи или мањи). Цеца зна да чоколада укупно има x редова, а део који јој је Неца показао има y редова.

Помозите Цеци и напишите јој колико редова има већи део чоколаде.

Опис улаза

Прва и једина линија стандардног улаза садржи два природна броја x и y ($1 \leq x < 100$, $1 \leq y < 100$) који представљају редом број редова целе чоколаде и број редова дела који је остао Цеци.

Опис излаза

На стандардни излаз исписати колико редова има већи део чоколаде.

Пример 1

Улаз	*Излаз*	*Објашњење*
`11 4`	`7`	Чоколада има 11 редова, а Цеци је остало 4. То значи да је Неца сакрио део који има 7 редова и то је уједно и већи део.

Пример 2

Улаз	*Излаз*	*Објашњење*
`5 3`	`3`	Чоколада има 5 редова, а Цеци је остало 3. То значи да је Неца сакрио део који има 2 реда, па је већи део онај који је остао Цеци.

Решење

Опис главног решења

Део чоколаде који види Цеца се уноси са улаза (y), а део чоколаде који је код Неце може да се израчуна као разлика целе чоколаде и Цециног дела (формула: $x - y$). Ако је Цецин део већи исписује се њен део, а ако није исписује се Нецин део.

```
cela, deo = map(int, input().split())

if cela - deo > deo:
    print(cela-deo)
else:
    print(deo)
```

Задатак: Јмбг

Аутор: Душан Попадић

Јединствени матични број грађана има 13 цифара подељених у 6 група. У овом задатку посматрамо само прве три групе, остале нам нису битне: - Прве две цифре одређују ког дана у месецу је особа рођена - Друге две цифре одређују месец у ком је особа рођена (01 - јануар, 02 - фебруар, …, 12 - децембар) - Наредне три цифре одређују годину када је особа рођена. Сматра се да је особа рођена између 1900. и 2025. године, па се прва цифра године подразумева.

На основу унетих првих 7 цифара матичног броја одредити колико особа има година на данашњи датум (15.11.2025).

Опис улаза

У првих 7 редова стандардног улаза се налази по једна од првих 6 цифара матичног броја. Гарантује се да ће унети подаци бити исправни (постоји унети датум и он није након 15.11.2025) и да ће прва цифра године бити или 9 или 0.

Опис излаза

У једином реду стандардног излаза исписати један број - колико особа има напуњених година данас. Ако је особи данас рођендан, сматра се да има онолико година колико данас пуни.

Пример 1

Улаз	*Излаз*	*Објашњење*

`2`
`9`
`1`
`2`
`9`
`9`
`5`
	`29`	Особа у питању је рођена 29. децембра 1995. године и данас има 29 година.

Пример 2

Улаз	*Излаз*	*Објашњење*

`1`
`4`
`0`
`6`
`0`
`0`
`7`
	`18`	Особа у питању је рођена 14. јуна 2007. године и данас има 18 година.

Пример 3

Улаз	*Излаз*	*Објашњење*

`1`
`8`
`0`
`1`
`0`
`2`
`5`
	`0`	Беба у питању је рођена 18. јануара 2025. године и данас има 0 напуњених година.

Пример 4

Улаз	*Излаз*	*Објашњење*

`1`
`5`
`1`
`1`
`0`
`0`
`0`
	`25`	Особа у питању је рођена 15. новембра 2000. године и данас пуни 25 година.

Решење

Прво је потребно да учитамо све неопходне податке са стандардног улаза. Олакшавајућа околност је што се информације о датуму рођења учитавају цифра по цифра, па лако можемо да реконструишемо датум. На пример, гледајући са лева на десно, ако је прва цифра дана рођења a и друга цифра дана рођења b, тада уз помоћ позиционог записа броја лако добијамо дан рођења према следећој формули $a * 10 + b$. На сличан начин можемо реконструисати месец и годину рођења, што препуштамо читаоцу. Дан, месец и годину рођења ћемо чувати редом у променљивама `dan`, `mesec`, `godina`.

Након реконструкције датума рођења, потребно је да израчунамо број година које корисник има дана 15.11.2025. Очигледно, број година корисника ће бити $g = 2025 - godina$. Међутим, морамо водити рачуна о дану и месецу рођења. Уколико је корисников рођендан после 15.11., јасно је да га још није прославио, па у том случају број година морамо умањити за један, тј. $g = 2025 - godina - 1$.

```python
dan = 0
mesec = 0
godina = 0

cifra = int(input())
dan = dan * 10 + cifra
cifra = int(input())
dan = dan * 10 + cifra

cifra = int(input())
mesec = mesec * 10 + cifra
cifra = int(input())
mesec = mesec * 10 + cifra

cifra = int(input())
godina = godina * 10 + cifra
cifra = int(input())
godina = godina * 10 + cifra
cifra = int(input())
godina = godina * 10 + cifra

if godina > 900:
    godina += 1000
else:
```

```
    godina += 2000

g = 2025 - godina

if mesec > 11 or (mesec == 11 and dan > 15):
    g -= 1

print(g)
```

Задатак: Комбинација задатака

Аутор: Филип Марић

У свету бројева води се турнир у ком се такмиче црвени и плави тим. Сваки тим има своје борце, представљене бројевима. Када се бирају парови који ће се надметати, борба може почети само ако борци **нису** исте снаге, односно ако њихови бројеви нису једнаки. Написати програм који исписује све могуће парове бораца за које може почети борба.

Опис улаза

- Први ред стандардног улаза садржи број c, други ред c различитих бројева (борци црвеног тима).
- Трећи ред улаза садржи број p, четврти p различитих бројева (борци плавог тима).

Опис излаза

Исписати све могуће парове за борбу, сваки у посебном реду. У сваком пару је потребно исписати прво број такмичара из црвеног тима, па број такмичара из плавог тима. Парове исписати у произвољном редоследу.

Пример

Улаз

```
3
5 1 3
3
1 4 3
```

Излаз

```
5 1
5 4
5 3
1 4
1 3
3 1
3 4
```

Објашњење

Такмичар са бројем 5 из црвеног тима може да се такмичи са свим такмичарима плавог тима, такмичар са бројем 1 из црвеног тима може да се такмичи са такмичарима 4 и 3 из плавог тима (не може против 1 јер имају исти број), а такмичар са бројем 3 из црвеног тима може да се такмичи са такмичарима 1 и 4 из плавог тима.

Решење

Опис главног решења

Прво треба учитати елементе, тј. бројеве црвеног тима, а затим бројеве плавог тима. Сачуваћемо их у векторима, тј. низовима који се редом зову c и p и имају nc и np елемената. Да бисмо решили задатак потребно је да извршимо пажљиву анализу текста и разумемо правила такмичења. Постоје два важна ограничења у тексту задатка:

1. У сваком тиму не постоје два борца исте снаге, тј. не постоје дупликати у низовима. Ово не значи да се у црвеном и плавом тиму не може наћи борац исте снаге.
2. Борба је могућа само између бораца различитих снага.

Одавде се лако може закључити следеће:

1. Борба се не може десити, само ако су борац из црвеног тима и борац из плавог тима исте снаге.
2. С обзиром да су сви елементи у низовима различити, то значи да се сваки борац из црвеног тима може борити са најмање $np - 1$ бораца из плавог тима, што одговара случају када у плавом тиму постоји борац чија је снага једнака снази црвеног борца. Са друге стране, сваки борац из црвеног тима се може борити са највише np, што одговара случају када у плавом тиму не постоји борац једнаке снаге снази црвено борца.

Додатно, одавде можемо закључити да у најгорем случају имамо $np \cdot nc$ могућих парова бораца које треба приказати. Управо овај закључак нам говори да задатак можемо да решимо једноставним упоређивањем свих могућих парова уз помоћ двоструке петље. Искуснији такмичар би могао помислити да постоји решење које је ефикасније од овога, али то није случај, јер се од нас очекује да прикажемо свих $np \cdot nc$ могућих парова, па нема сврхе дизајнирати ефикасније решење.

```
nc = int(input())
crveni = [int(ul) for ul in input().split(' ')]
np = int(input())
plavi = [int(ul) for ul in input().split(' ')]

for i in range(nc):
    for j in range(np):
        if crveni[i] != plavi[j]:
            print(crveni[i], ' ', plavi[j])
```

Задатак: Број сегмената парног збира

Аутор: Филип Марић

Криптографски систем посматра све сегменте датог низа битова (тј. бројева 0 и 1). Ако је збир свих битова у неком сегменту паран, систем га сматра валидним. Твој задатак је да за дати низ битова одредиш колико валидних сегмената постоји.

Напомена: Сегмент низа је било који део низа који садржи узастопне елементе и није празан. На пример, сегменти низа $[1, 2, 3]$ су $[1]$, $[1, 2]$, $[1, 2, 3]$, $[2]$, $[2, 3]$ и $[3]$.

Опис улаза

У првом реду стандардног улаза налази се цео број n $(1 \leq n \leq 5 \cdot 10^4)$.

У другом реду стандардног улаза налази се n размаком раздвојених целих бројева $a_1, a_2, \ldots, a_n$ (ови бројеви су 0 или 1) - представљају чланове низа.

*Додатна ограничења*Тест примери су подељени у две групе: - У тест примерима вредним 60 поена важи $n \leq 100$; - У тест примерима вредним 40 нема додатних ограничења.

Опис излаза

На стандардни излаз исписати тражени број валидних сегмената.

Пример

Улаз	*Излаз*	*Објашњење*
`5` `1 0 1 1 0`	`7`	Сегменти парног збира су $[1, 0, 1]$, $[0]$, $[0, 1, 1]$, $[0, 1, 1, 0]$, $[1, 1]$, $[1, 1, 0]$ и $[0]$.

Решење

Опис наивног решења

Да бисмо решили задатак потребно је да прво учитамо дати низ a нула и јединица. Наивно, можемо да испитамо сваки могући сегмент, тј. подниз узастопних елемената и да израчунамо његов збир.

```
# Učitavanje ulaza
n = int(input())
a = list(map(int, input().split()))

broj = 0
for i in range(n):
    zbir = 0
    for j in range(i, n):
        zbir += a[j]
        if zbir % 2 == 0:
            broj += 1

print(broj)
```

Опис главног решења

Наивно решење је једноставно, али неефикасно за велике низове, па је потребно да размислимо како можемо да побољшамо ефикасност. Ефикасност наивног решења је квадратна по броју елемената низа, тј. $O(n^2)$. Да бисмо то постигли потребно је да кренемо од дефиниције проблема који решавамо и да уочимо да ли постоји неко правило које можемо једноставно да искористимо.

С обзиром да треба да одредимо колико има сегмената чији је збир паран, могли бисмо да кренемо од прецизног дефинисања када сегмент има паран збир. Да би сегмент имао паран збир, он мора имати паран број јединица. Одавде можемо да закључимо да нам треба ефикасан начин израчунавања збирова сегмената. То можемо лако постићи, ако одржавамо префиксне збирове нашег низа. Прецизније, сегмент низа $a[i, \dots, j]$ имаће паран збир само ако су префикси низа $a[0, \dots, i]$ и $a[0, \dots, j]$ исте парности. Прецизније, број парних сегмената једнак је броју префикса са истом парношћу.

Због свега наведеног у нашем решењу ћемо одржавати следеће вредности:

- `ps` - текућа префиксна сума.
- `broj` - број сегмената исте парности.
- `brojParnihPS` - број сегмената са парном префиксном сумом.
- `brojNeparnihPS` - број сегмената са непарном префиксном сумом.

Префиксне суме можемо да рачунамо инкрементално и да током рачунања префиксних сума истовремено одржавамо и вредности осталих променљивих. Као и у сваком инкременталном решењу, крећемо од празног сегмента и у свакој итерацији текући сегмент проширујемо следећим елементом низа. Приметимо да то проширење може бити елементом 0 или 1. Уколико проширујемо елементом 0, парност текуће префиксне се неће променити, док проширивањем са 1 мењамо парност текуће префиксне суме. Дакле, ако је нова префиксна сума парна, тада нови елемент формира сегменте парног збира са свим претходним парним сегментима. Ако је нова префиксна сума непарна, тада нови елемент формира сегменте парног збира са свим претходним непарним сегментима.

Да би читаоцу била јасна ова идеја, размотрићемо корак по корак израчунавање префиксних сума на примеру низа из текста задатка. Нека је дат низ $1, 0, 1, 1, 0$. Редослед итерација и вредности променљивих можемо да прикажемо следећом таблицом.

i	a[i] (елемент којим проширујемо префикс)	ps (нова вредност префиксне суме)	Парност префикса	broj (текући број сегмената парног збира)	brojParnihPS	brojNeparnihPS
0	1	1	непаран	број += brojNeparnihPS = 0 → 0	1	1
1	0	1⊕0=1	непаран	број += brojNeparnihPS = 1 → 1	1	2
2	1	1⊕1=2	паран	број += brojParnihPS = 1 → 2	2	2
3	1	2⊕1=3	непаран	број += brojNeparnihPS = 2 → 4	2	3
4	0	3⊕0=3	непаран	број += brojNeparnihPS = 3 → 7	2	4

Ради једнставности довољно је да чувамо само парност префиксне суме, јер од парности зависи да ли формирамо сегменте са парним или непарним претходнисм сегментима. Приметимо да је ефикасно решење линеарне сложености, тј. $O(n)$. Решење је у наставку.

```
n = int(input())
a = list(map(int, input().split()))

broj = 0
prefix_suma = 0
broj_parnih = 1
broj_neparnih = 0

for i in range(n):
    prefix_suma = (prefix_suma + a[i]) % 2
    if prefix_suma % 2 == 0:
```

```
        broj += broj_parnih
        broj_parnih += 1
    else:
        broj += broj_neparnih
        broj_neparnih += 1

print(broj)
```

Задатак: Контролна цифра

Аутор: Филип Марић

Идентификациони бројеви (нпр. ISBN, JMBG, бројеви банковних рачуна и кредитних картица) се обично штите од грешака увођењем тзв. контролне цифре. Последња цифра броја се одређује тако да се нека израчуната статистика свих цифара броја буде дељива неким бројем. На пример, четвороцифрени број $ABCD$ можемо заштитити контролном цифром X тако што ћемо X одредити тако да у добијеном броју $ABCDX$ важи да је $A + 2B + 3C + 4D + X$ дељиво бројем 10. Написати програм који за унети четвороцифрени број $ABCD$ одређује најмању вредност конторлне цифре X.

Опис улаза

Са стандардног улаза се учитава четвороцифрени број $ABCD$ (који евентуално може да има и водеће нуле).

Опис излаза

На стандардни излаз исписати контролну цифру X.

Пример 1

Улаз	*Излаз*	*Објашњење*
`1234`	`0`	Важи да је $1 + 2 \cdot 2 + 3 \cdot 3 + 4 \cdot 4 = 30$, који је већ дељив са 10. Зато је најмањи број X који се може додати на овај број да би он постао дељив са 10 број $X = 0$.

Пример 2

Улаз	*Излаз*	*Објашњење*
`8352`	`3`	Важи да је $8 + 2 \cdot 3 + 3 \cdot 5 + 4 \cdot 2 = 37$. Најмањи број X који се може додати на овај број да би он постао дељив са 10 је 3.

Решење

Учитавамо број и одређујемо му појединачне цифре A, B, C и D. Након тога можемо израчунати вредност израза $A + 2B + 3C + 4D$. На њега треба додати неку вредност X тако да збир буде дељив са 10 тј. да му је последња цифра нула. Ако је последња цифра збира $A+2B+3C+4D$ нека цифра k, тада се може додати број $X = 10-k$. Ово је и коначно решење у свим случајевима осим када је $k = 0$, јер је тада уместо $X = 10$ могуће додати $X = 0$. Зато је пре исписа коначног резултата потребно још одредити остатак при дељењу X са 10 (или гранањем обрадити овај специјални случај).

```
# odredjujemo pojedinacne cifre broja
broj = int(input())
```

```
A = (broj // 1000) % 10
B = (broj // 100) % 10
C = (broj // 10) % 10
D = (broj // 1) % 10

# odredjujemo kontrolnu cifru - najmanju cifru X tako da
# A+2B+3C+4D+X bude deljivo sa 10
X = (10 - (A + 2*B + 3*C + 4*D) % 10) % 10

print(X)
```

Задатак: Најубедљивија победа

Аутор: Филип Марић

Одиграно је коло у кошаркашкој лиги и познати су резултати. Напиши програм који одређује највећу разлику којом је неки тим победио свог противника.

Опис улаза

Са стандардног улаза се уноси природан број n ($1 \leq n \leq 32$), а затим n парова целих бројева који представљају резултате n утакмица.

Опис излаза

На стандардни излаз исписати највећу разлику.

Пример

Улаз	*Излаз*	*Објашњење*
9 98 91 88 63 62 76 96 86 99 75 91 73 88 79 80 69 63 88	25	Најубедљивију победу су остварили тимови који су своје противнике добили резултатом 88:63 (сасвим случајно, то се у овом колу догодило два пута).

Решење

Задатак се своди на уобичајено одређивање максимума низа бројева. Разлике у поенима морају бити природни бројеви, па на старту можемо да претпоставимо да је максимална разлика 0. Током учитавања бројева у петљи, одредићемо њихову апсолутну разлику и упоредити је са текућим максимумом. Ако је текућа разлика већа од максимума, онда ће то бити нови максимум. На крају, потребно је исписати вредност максимума коју смо одредили.

```
n = int(input())
max_razlika = 0
for i in range(n):
    a, b = map(int, input().split())
```

```
    max_razlika = max(max_razlika, abs(a - b))
print(max_razlika)
```

Задатак: Јединичне колоне

Аутори: Милан Вугделија, Душан Поповић

Дата је матрица бројева димензија $n \times m$ која садржи само нуле и јединице. Написати програм који исписује колико постоји колона у матрици у којима је број нула највише 2.

Опис улаза

У првом реду се налазе два броја n и m ($2 \leq n, m \leq 100$) који представљају димензије матрице. У наредних n редова се налази по m бројева раздвојених размацима (сви су или 0 или 1). Ови бројеви представљају дату матрицу.

Опис излаза

У једином реду стандардног излаза исписати тражени број колона.

Пример

Улаз	*Излаз*	*Објашњење*

`4 5`
`1 0 1 1 0`
`1 1 0 1 0`
`1 0 0 1 1`
`1 0 0 0 1`
	`3`	Прва, четрвртa и пета колона имају највише две нуле.

Решење

Да бисмо одредили број колона у којима нема више од две нуле потребно је да кроз матрицу прођемо по колонама и за сваку колону изборојимо колико има нула. Уколико је тај број мањи или једнак 2, повећавамо бројач. На крају исписујемо вредност бројача.

```
n, m = map(int, input().split())

mat = []
for _ in range(n):
    row = list(map(int, input().split()))
    mat.append(row)

brk = 0

for j in range(m):
    brn = 0
    for i in range(n):
        if mat[i][j] == 0:
            brn += 1
    if brn <= 2:
        brk += 1

print(brk)
```

Задатак: Постојање троугла

Аутор: Љубомир Бановић

Дата је гомила на којој се налази n штапића. Потребно је одредити да ли постоје три различита штапића са гомиле, таква да формирају троугао. Штапићи дужине a, b и c формирају троугао ако важе следеће неједнакости: $a + b > c$ $b + c > a$ $c + a > b$

Опис улаза

У првом линији стандардног улаза се налази један ненегативан цео број n ($1 \leq n \leq 10^5$).

У другој линији стандардног улаза се налази n ненегативних целих бројева: $a_1, a_2, ... a_n$ ($1 \leq a_i \leq 10^9$) који представљају дужину штапића на гомили.

*Додатна ограничења*Тест примери су подељени у две групе: - У тест примерима вредним 30 поена важи $n \leq 100$; - У тест примерима вредним 70 поена нема додатних ограничења.

Опис излаза

На стандардни излаз потребно је исписати `da` ако постоје такви штапићи, а у супротном исписати `ne`.

Пример 1

Улаз	*Излаз*
`5` `23 2 10 3 11`	`da`

Пример 2

Улаз	*Излаз*
`3` `3 27 10`	`ne`

Решење

Наивно решење

Задатак можемо решити директно тако што у трострукој `for` петљи упоредимо све могуће тројке штапића и испитамо да ли испуњавају услов. Решење је идејно лако, али је неефикасно за велико n, јер је временска сложеност овог решења $O(n^3)$. Ово решење доноси 30 поена.

```
n = int(input())
v = list(map(int, input().split()))

for i in range(n):
    for j in range(i + 1, n):
        for k in range(j + 1, n):
            if v[i] + v[j] > v[k] and v[i] + v[k] > v[j] and v[k] + v[j] > v[i]:
                print("da")
                exit()

print("ne")
```

Решење соритрањем и посматрањем узастопних бројева

Да би наш програм радио довољно брзо за велике вредности n, потребно је да осмислимо ефикаснији алгоритам. Потребно је да осмислимо решење које неће испитивати све могуће тројке штапића. Кренућемо од неједнакости троугла која каже да се од дужи чије су дужине редом a, b и c може формирати троугао само ако важе следеће неједнакости:

- $a + b > c$
- $b + c > a$
- $c + a > b$

Ако замислимо како изгледа разнострани троугао, ове услове можемо мало да релаксирамо. Прецизније, није потребно испитивати све три неједнакости, ако одредимо најдужу страницу троугла. Да бисмо формирали троугао довољно је да дужина најдуже страница троугла буде мања од збира дужина преостале две странице. Дакле, за сваки штапић v_i треба да одредимо два штапића v_j, v_k таква да су њихове дужине мање или једнаке од дужине v_i, тј. мора важити $v_j \leq v_k \leq v_i$. У том случају, троугао бисмо могли да формирамо ако би важило $v_j + v_k > v_i$.

Да би наш алгоритам био довољно брз, потребно је да ову претрагу учинимо што је могуће ефикаснијом. Један начин да то урадимо, јесте да уочимо да нама не требају било какви штапићи v_j, v_k за које важи $v_j + v_k > v_i$, већ нам требају они v_j, v_k за које постоји највећа шанса да буде испуњена неједнакост $v_j + v_k > v_i$. Лако се уочава да то морају бити највећи могући v_j, v_k који су мањи од v_i.

Сортираћемо наш низ неопадајуће и провераваћемо редом да ли тројке елемената v_{i-2}, v_{i-1}, v_i задовољавају неједнакост $v_{i-2} + v_{i-1} > v_i$. Чим наиђемо на такву тројку, то значи да од датих штапића можемо формирати троугао. Ако не нађемо тројку која испуњава дату неједнакост, онда не можемо формирати троугао.

Временска сложеност ефикасног решења је $O(n \cdot log(n))$.

```python
n = int(input())
v = list(map(int, input().split()))

v.sort()

for i in range(2, n):
    if v[i - 2] + v[i - 1] > v[i]:
        print("da")
        exit()

print("ne")
```

Решење засновано на Фибоначијевом низу

Постоји још један начин на који може да се реши овај задатак. Користићемо резултат из претходног решења: ако је низ сортиран и не садржи троугао, за сваки индекс i важи: $v_i + v_{i+1} \leq v_{i+2}$.

Конструишимо најдужи сортирани (растући) низ без троуглова. При конструкцији овог низа, морамо да пазимо на оригинално ограничење задатка, да је сваки члан низа природан број мањи или једнак 10^9. За прва два члана низа бирамо две јединице (најмања могућа дужина штапића), а за следеће чланове низа бирамо штапић такав да је једнак дужини збира претходна два (дужина овог штапића мора да испуњава неједнакост, а бирамо најмању такву вредност). Овим приступом смо добили низ $1, 1, 2, 3, 5, 8, 13...$

Ако мало боље обратимо пажњу, можемо да приметимо да је ово заправо Фибоначијев низ! Како Фибоначијев низ расте брзо (експоненционално, 45. члан Фибоначијевог низа је већи од 10^9), закључујемо да сваки низ дужине веће од 44 мора да садржи троугао, јер ће у супротном постојати индекс за који не важи неједнакост. Решење овим приступом би подразумевао испис “da” за сваки низ величине веће од 44 и употреба наивног кубног решења за низове мање или једнаке дужини 44.

Временска сложеност овог решења је $O(n^3)$ ако је $n < 100$ иначе је $O(1)$. Дакле, колико год да је n овај програм извршава највише милион (100^3) итерација, па можемо рећи да има константну сложеност.

```
n = int(input())
v = list(map(int, input().split()))

if n >= 100:
    print("da")
    exit()

for i in range(n):
    for j in range(i + 1, n):
        for k in range(j + 1, n):
            if v[i] + v[j] > v[k] and v[i] + v[k] > v[j] and v[k] + v[j] > v[i]:
                print("da")
                exit()

print("ne")
```

Задатак: Аутопревозник

Аутори: Нина Икодиновић, Душан Попадић

На путу Београд - Крагујевац постоји N аутобуских станица. На свакој од тих станица путник може да купи аутобуску карту до било које друге станице на том путу. Колико различитих карата постоји у продаји тако да су и почетна и крајња станица на путу Београд - Крагујевац?

Опис улаза

У првом и јединому реду се уноси природни број N ($2 \leq N \leq 100$), број аутобуских станица на релацији Београд - Крагујевац.

Опис излаза

Излаз је једна вредност која представља број различитих карата које постоје у продаји.

Пример 1

Улаз	*Излаз*	*Објашњење*
3	6	Рецимо да постоје 3 станице: Београд, Аранђеловац и Крагујевац. Различитих карата има 6: Београд - Аранђеловац, Београд - Крагујевац, Аранђеловац - Београд, Аранђеловац - Крагујевац, Крагујевац - Београд и Крагујевац - Аранђеловац.

Пример 2

Улаз	*Излаз*
5	20

Пример 3

Улаз	*Излаз*
67	4422

Решење

Задатак се врло лако може решити пажљивом анализом проблема. Ако је дато n станица дуж пута, јасно је да се из сваке појединачне станице могу купити карте до преосталих $n-1$ станица. С обзиром да станица дуж пута има n, укупан број карата које се могу купити је $n*(n-1)$. Решење је у наставку.

```
n = int(input())
print(n * (n - 1))
```

Задатак: Термометар

Аутор: Огњен Тешић

Познате су минимална и максимална температура за данашњи дан. Термометар је, међутим, покварен, па не показује увек стварну температуру.

Потребно је написати програм који на основу унете вредности температуре T, коју показује термометар, исписује једну од следећих порука: - `ISPOD MINIMUMA` - ако је T мања од минималне температуре A; - `IZNAD MAKSIMUMA` - ако је T већа од максималне температуре B; - `NA GRANICI` - ако је T једнака A или B; - `IZMEDJU` - ако је $A < T < B$.

Опис улаза

Једина линија стандардног улаза садржи три цела броја раздвојена размаком, A, B, T $(-10 \leq A, B, T \leq 40,\ A < B)$ који представљају, редом, минималну температуру на данашњи дан, максималну температуру на данашњи дан и температуру коју термометар показује.

Опис излаза

Исписати једну од следећих порука: `ISPOD MINIMUMA`, `IZNAD MAKSIMUMA`, `NA GRANICI` или `IZMEDJU`. Важно је да порука коју испишеш буде идентична као једна од наведених (исте речи и велика слова).

Пример 1

Улаз	*Излаз*	*Објашњење*
`5 15 12`	`IZMEDJU`	*Објашњење.* Минимална температура за данашњи дан је 5, а максимална 15. Термометар показује 12, па је вредност између минималне и максималне температуре.

Пример 2

Улаз	*Излаз*
`6 14 20`	`IZNAD MAKSIMUMA`

Пример 3

Улаз	*Излаз*
`4 16 4`	`NA GRANICI`

Пример 4

Улаз	*Излаз*
`-5 3 -10`	`ISPOD MINIMUMA`

Решење

Овај задатак представља класичну проверу припадности броја задазим опсезима који се решава уланчавањем неколико `if - else` блокова.

```
A, B, T = map(int, input().split())
```

```
if T < A:
    print("ISPOD MINIMUMA")
elif T > B:
    print("IZNAD MAKSIMUMA")
elif T == A or T == B:
    print("NA GRANICI")
else:
    print("IZMEDJU")
```

Задатак: Једначина

Аутор: Александар Николић

Дат је низ $a_1, a_2, \dots, a_n$ и број x. Одредити број парова индекса (i, j) таквих да је $i < j$ и важи $a_i + a_j = x$, као и $a_i - a_j = x$.

Опис улаза

Први ред стандардног улаза садржи два цела броја n и x - представљају број чланова низа и број x.

Други ред стандардног улаза садржи n размаком раздвојених целих бројева $a_1, a_2, \dots, a_n$ - представљају чланове низа.

Опис излаза

Један број - укупан број парова.

Ограничења- $1 \leq n \leq 200.000$ - $-10^9 \leq x \leq 10^9$

Тест примери су подељени у три групе:

- у тест примерима вредним 10 поена важи: $x = 0$;
- у тест примерима вредним 40 поена важи: $n \leq 1000$;
- у тест примерима вредним 50 поена нема додатних ограничења.

Пример

Улаз	*Излаз*	*Објашњење*
`5 2` `2 0 3 0 2`	`2`	*Објашњење.* Парови чији су индекси $(1, 2)$ и $(1, 4)$ испуњавају услов задатка (нумерација почиње од 1, не од 0).

Решење

Наивно решење

Најједноставније је овом задатку приступити директно: пролазимо кроз све парове и проверавамо да ли су испуњени задати услови. Ово се једноставно ради са две угнежђене петље и једном провером услова. Сложеност овог решења је $O(n^2)$ и оно доноси 40 поена.

```
n, x = map(int, input().split())
a = list(map(int, input().split()))

resenje = 0
```

```
for i in range(n):
    for j in range(n):
        if a[i] + a[j] == x and a[i] - a[j] == x:
            resenje += 1

print(resenje)
```

Оптимално решење

Хајде да мало боље погледамо услове који су нам дати. Пошто важи $a_i + a_j = x$ и $a_i - a_j = x$ онда мора важити и $a_i + a_j = a_i - a_j$, а одатле очигледно следи $a_j = 0$. Када у једну од прве две једначине заменимо вредност 0 за a_j добијамо $a_i = x$. Дакле, поставља се питање колико има парова таквих да је $a_i = x$ и $a_j = 0$, уз додатни услов $i < j$, тј. 0 се мора наћи после x. Ако бисмо исписали на папир све парове, свако x би се појавило на папиру онолико пута колико иза њега има нула, па укупан број парова можемо да добијемо тако што за свако x на неки збир додамо број нула који се налази иза њега. Ово можемо урадити у једном пролазу кроз низ од краја ка почетку. Док пролазимо кроз низ бројимо колико пута се појављује 0, а кад наиђемо на x, на неки збир додамо број нула који смо до тог тренутка пребројали.

```
n, x = map(int, input().split())
a = list(map(int, input().split()))

resenje = 0
broj_nula = 0

for i in range(n - 1, -1, -1):
    if a[i] == x:
        resenje += broj_nula
    if a[i] == 0:
        broj_nula += 1

print(resenje)
```

Задатак: Највећа комбинација

Аутор: Андреј Павловић

Дато је низ A од N бројева мањих од 10^{18}. Потребно их је спојити у један број тако да тај број буде највећи могући. Под спајањем се сматра надовезивање бројева без мењања поретка цифара у сваком од бројева.

Опис улаза

Уноси се број $N \leq 10^5$ и низ A од N бројева. Важи да је $1 \leq A_i \leq 10^{18}$ за све $1 \leq i \leq N$.

Опис излаза

Један број - највећи број који се добија спајањем низа бројева у један велики број.

Ограничења- $1 \leq N \leq 10^5$ - $1 \leq A_i \leq 10^{18}$

Тест примери су подељени у три групе:

- у тест примерима вредним 10 поена важи: $N \leq 3$;
- у тест примерима вредним 30 поена важи: $N \leq 10$;
- у тест примерима вредним 60 поена нема додатних ограничења.

Пример 1		**Пример 2**	
Улаз	*Излаз*	*Улаз*	*Излаз*
`2`	`19190`	`3`	`91512`
`190 19`		`15 12 9`	

Решење

Опис главног решења

Бројеве је згодно посматрати као ниске, јер се под „спајањем” подразумева надовезивање њихових цифара. Зато сваки број претварамо у ниску и затим одређујемо њихов најбољи редослед.

За два броја представљена као ниске `x` и `y`, разматрамо две могућности: да прво стоји `x`, па `y`, или обрнуто. Упоређујемо ниске `x + y` и `y + x`, где знак `+` означава конкатенацију (надовезивање ниски). Ако је `x + y` лексикографски веће од `y + x`, онда је боље да `x` стоји испред `y`.

На основу овог правила сортирамо све ниске, а затим их редом надовежемо. Тако добијамо један велики број који је највећи могући број који се може добити спајањем датих бројева.

Глава 2

2. круг квалификација

Задатак: Гринготс

Аутор: Душан Попадић

У магијском свету Харија Потера се за плаћање користе галеони, сикли и кнути. Један галеон има 17 сикла, а један сикл 29 кнута. Рон има код себе a галеона, b сикла и c кнута и жели да их раситни тако да има само кнуте. Он одлази у чаробњачку банку Гринготс и тамо раситњава новац. Колико Рон има кнута након раситњавања?

Опис улаза

У првом реду стандардног улаза се налази број a $(0 \leq a \leq 100)$. У другом реду стандардног улаза се налази број b $(0 \leq b \leq 100)$. У трећем реду стандардног улаза се налази број c $(0 \leq c \leq 100)$.

Опис излаза

У јединoм реду стандардног излаза исписати колико ће Рон имати кнута након уситњавања.

Пример 1

Улаз	*Излаз*	*Објашњење*

`5`
`26`
`30`
	`3249`	Након уситњавања када све галеоне и сикле претвори у кнуте, Рон ће имати 3249 кнута.

Пример 2

Улаз	*Излаз*

`0`
`1`
`1`
	`30`

Решење

Опис главног решења

Потребно је све износе свести на кнуте. Један галеон вреди $17 \cdot 29$ кнута, а један сикл 29 кнута, па укупан број кнута добијамо као

$a \cdot 17 \cdot 29 + b \cdot 29 + c$. Ова вредност се директно израчуна и испише као резултат.

```
a = int(input())
b = int(input())
c = int(input())

knuti = a * 17 * 29 + b * 29 + c;
print(knuti)
```

Задатак: Путовање

Четворо другара Марина, Нађа, Влад и Петар отишли су на тродневно путовање заједно. Сваки дан они су сели у неки кафић и попили сви по једно исто пиће. Договорили су се да ће сваки дан неко други платити рачун за све, а да ће трошкове поделити на крају путовања. Како не воли много да плаћа, Влад није ни један дан платио рачун у кафићу за све другаре. Због тога, остали међу собом збијају шале на Владов рачун. Како би могли више шала да направе, занима их коме од њих Влад дугује највише. Помозите им у томе.

Опис улаза

У првом реду стандардног улаза налази се један позитиван реалан број – износ рачуна који је платила Марина. У другом реду стандардног улаза налази се један позитиван реалан број – износ рачуна који је платила Нађа. У трећем реду стандардног улаза налази се један позитиван реалан број – износ рачуна који је платио Петар. Ниједан од ових бројева није већи од 100 и сви су различити.

Опис излаза

У јединому реду стандардног излаза исписати прво слово имена другара коме Влад дугује највише новца (`M`, `N` или `P`).

Пример 1

Улаз	*Излаз*

`10.92`
`14.32`
`13.75`
	`N`

Пример 2

Улаз	*Излаз*

`18.5`
`12.45`
`17.8`
	`M`

Решење

Опис главног решења

Како у сваком кафићу сви другари пију иста пића, Влад дугује свакоме по четвртину износа рачуна. Самим тим, Влад дугује највише оном другару који је платио највећи износ рачуна.

Задатак: Приближан рачун

Аутор: Милан Вугделија

Ана жели да брзо процени вредност ствари које је ставила у колица за куповину. Она не сабира тачне цене, већ вредности заокругљене на најближу стотину. У случају нееднозначности (ако су две стотине једнако близу), Ана заокругљује цену на вишу стотину. Написати програм који учитава број купљених ствари n и цене тих ствари, а исписује тачан збир и збир који је добила Ана својим поступком.

Опис улаза

У првом реду стандардног улаза је природан број n, не већи од 20. У другом реду је n природних бројева, не већих од 10000, раздвојених по једним размаком.

Опис излаза

На стандардни излаз исписати само два цела броја, сваки у посебном реду. Први број је тачна укупна вредност ствари у колицима, а други број је приближан збир који је Ана добила.

Пример 1

Улаз	*Излаз*	*Објашњење*
`5` `189 34 450 999 300`	`1972` `2000`	Тачан збир је: 189 + 34 + 450 + 999 + 300 = 1972. Ана је сабирала редом: 200 (уместо 189) + 0 (34) + 500 (450) + 1000 (999) + 300 (300) = 2000.

Пример 2

Улаз	*Излаз*
`3` `249 349 199`	`797` `700`

Решење

Кључни део решења је формула по којој се израчунава приближна вредност y дате ствари, ако је њена цена x. Заокругљивање на ближу стотину може да се изведе додавањем 50 на цену, а затим заокругљивањем на нижу стотину.

$$y = \left\lfloor \frac{x + 50}{100} \right\rfloor \cdot 100$$

При томе, у случају да се цена завршава на 50 и једнако је удаљена од више и ниже стотине, на овај начин добијамо цену заокругљену на вишу стотину, као што се и тражи. На пример, за $x = 150$ добијамо

$$y = \left\lfloor \frac{150 + 50}{100} \right\rfloor \cdot 100 = \lfloor 2 \rfloor \cdot 100 = 2 \cdot 100 = 200.$$

Остаје само да се ова формула примени на сваку ствар из колица и да се добијене вредности саберу.

```python
n = int(input())
cena = list(map(int, input().split()))
zbir = 0
priblizno = 0
for i in range(n):
    zbir += cena[i]
    priblizno += (cena[i] + 50) // 100 * 100

print(zbir)
print(priblizno)
```

Задатак: Добри парови

Аутор: Огњен Тешић

Дат је цео број n. Исписати све парове бројева (i, j) за које важи $i, j \in \{1, 2, \ldots, n\}, i > j$ и $i + j$ је дељив са 3.

Опис улаза

Са стандардног улаза се учитава цео број n ($1 \le n \le 200$).

Опис излаза

Исписати све парове (i, j) који задовољавају услове. Сваки пар исписати у посебном реду, тако да се најпре испише број i, а затим број j. Редослед редова у излазу није битан. Ако нема ниједног пара, исписати -1.

Пример

Улаз	*Излаз*

`6`	`2 1`
	`4 2`
	`5 1`
	`5 4`
	`6 3`

Решење

Опис главног решења

Најпре примећујемо да су ограничења мала ($n \le 200$), па можемо без проблема да испитамо све могуће парове бројева (i, j). Уведимо логичку променљиву `ima_par` коју иницијално постављамо на `false`. Она ће нам служити да запамтимо да ли је током рада програма пронађен бар један пар који задовољава услове задатка.

Затим користимо две угнеждене петље. Спољашња петља пролази кроз све вредности броја i од 1 до n, док унутрашња петља пролази кроз све вредности броја j од 1 до $i - 1$. На овај начин аутоматски обезбеђујемо услов $i > j$, па нема потребе да га додатно проверавамо.

Унутар унутрашње петље проверавамо да ли је збир $i + j$ дељив са 3, односно да ли важи услов $(i + j) \bmod 3 = 0$. Ако је услов испуњен, пар (i, j) одмах исписујемо у излаз и постављамо променљиву `ima_par` на `true`, чиме бележимо да је пронађен бар један одговарајући пар.

Након што се све могуће комбинације бројева i и j обраде, проверавамо вредност променљиве `ima_par`. Уколико ниједан пар није пронађен, односно ако је `ima_par` остала `false`, на излаз исписујемо број -1.

Редослед у коме се парови исписују није битан, па је исправно исписивати их у тренутку када се пронађу.

Задатак: Цензура

Аутор: Љубомир Булатовић

Влада Републике Рохан је суочена са проблемом - огромном заступљеношћу недозвољених речи у новинама. Зато вас премијер моли да напишете програм који ће цензурисати неприкладна слова у датој речи, тако што ће свако недозвољено слово заменити симболом `'#'`.

Опис улаза

У првом реду стандардног улаза се налазе два цела броја n и k ($1 \leq n \leq 10^4, 1 \leq k \leq 26$)- дужина речи и број неприкладних слова.

У другом реду стандардног улаза се налази ниска дужине n, састављена од малих слова енглеске абецеде - реч коју је потребно цензурисати.

У трећем реду стандардног улаза се налази k различитих малих слова енглеске абецеде - неприкладна слова.

Опис излаза

У првом реду стандардног излаза исписати једну ниску - цензурисану реч.

Пример 1

Улаз	*Излаз*

`8 3`
`abcdefgh`
`c f h`
	`ab#de#g#`

Пример 2

Улаз	*Излаз*

`5 1`
`aaaaa`
`b`
	`aaaaa`

Решење

Опис главног решења

Најпре се учитава реч коју је потребно цензурисати, као и скуп неприкладних слова. Неприкладна слова се смештају у скуп. Затим се пролази кроз сваки карактер речи и, уколико се он налази у скупу недозвољених слова, исписује се знак `#`, а у супротном се исписује оригинално слово.

Задатак: Ширење тајне

Аутор: Теодора Обрадовић

Милан много воли да шири туђе тајне у школи. Јутрос је пре часова сазнао једну веома важну тајну и жели да сви сазнају за њу. Одлучио је да свима каже тајну у току часа. На часу сви ученици седе у једном реду. Милан ће да каже тајну ученику лево од себе(ако постоји) и ученику десно од себе (ако постоји). Затим ће ученик који седи лево од њега да каже ученику који седи још једно место у лево, а ученик који седи десно од њега ће рећи ученику који седи још једно место у десно. Ово ће се понављати све док се тајна не прошири до крајева реда. Међутим, постоје ученици који прате час и које не занимају туђе тајне. Они неће даље ширити тајну (иако можда не седе на крају реда).

Наставници се не свиђа што ученици причају на часу, па је направила неколико распореда седења. У свим распоредима седења ученици који прате наставу седе на истим местима, јер они не праве буку и њих не мора да премешта. Одредите колико ученика ће да сазна тајну за сваки распоред седења ако је познато где је наставница рекла Милану да седи.

Опис улаза

Прва линија стандардног улаза садржи природан број $n \leqslant 10^9$ који представља укупан број ученика.

Друга линија стандардног улаза садржи природан број m ($m \leqslant n$ и $m \leqslant 10^5$) који представља број ученика који прате наставу.

У трећој линији стандардног улаза уноси се m бројева између 1 и n - редни бројеви места где седе ученици који прате наставу.

У четвртој линији стандардног улаза уноси се број $r \leqslant 10^5$ - број распореда који је смислила наставница.

У наредних r линија стандардног улаза уноси се број између 1 и n - редни број места где у распореду седи Милан. Гарантује се да на овом месту не седи ученик који прати наставу из друге линије улаза.

Опис излаза

На стандардни излаз у r редова исписати колико ученика је сазнало тајну.

Подзадаци

- $n \leqslant 1000$ и $r \leqslant 1000$ - 20 поена;
- $m \leqslant 1000$ и $r \leqslant 1000$ - 40 поена;
- без додатних ограничења - 40 поена.

Пример 1

Улаз	*Излаз*	*Објашњење*

`10`
`3`
`1 4 7`
`2`
`2`
`9`
	`2`
	`3`
		У првом распореду седења за тајну ће сазнати ученици који седе на местима 2 и 3. Ученици 1 и 4 прате на часу и неће даље ширити ову тајну. У другом распореду седења ученици 8,9 и 10 ће сазнати за тајну. Ученик 7 прати на часу, а ученик 10 седи на крају реда и нема коме другом да је каже.

Пример 2

Улаз	*Излаз*	*Објашњење*

`1000`
`3`
`100 255 932`
`2`
`400`
`934`
	`676`
	`68`
		У првом распореду седења за тајну ће сазнати ученици који седе на местима која имају редне бројеве од 256 до 931 (укључујући та два). Ученици 255 и 932 прате на часу и неће даље ширити ову тајну. У другом распореду седења ученици који седе лево од ученика 932 ће сазнати за тајну. Ученик 932 прати на часу, а ученик 1000 седи на крају реда и нема коме другом да је каже.

Решење

Опис решења првог подзадатка

Када су и n и r мањи или једнаки 1000, задатак се може решити грубом силом. Довољно је да сваки пут симулирамо ширење тајне почевши од Милановог места и избројимо до колико ученика је стигла тајна.

```
n=int(input())
m=int(input())
prate=[]
for i in range(n+1):
    prate.append(0)
a=input().split()
```

```
for i in range(m):
    prate[int(a[i])]=1;
r=int(input())
while r>0:
    milan=int(input())
    res=1
    i=milan
    while i>1 and prate[i-1]==0:
        i-=1
        res+=1
    i=milan
    while i<n and prate[i+1]==0:
        i+=1
        res+=1
    print(res)
    r-=1
```

Опис решења другог подзадатка

Применом грубе силе решење неће да се изврши довољно брзо, због тога што је n веома велики број. Међутим, када су и m и r мањи или једнаки 1000, задатак се може решити на следећи начин: Приметимо да ће тајну да сазнају сви ученици између краја реда и неког ученика који прати наставу или два ученика који прате наставу. То је исто као да на позицијама 0 и $n + 1$ седе још два ученика који прате наставу и да тајну сазнају сви који седе између два ученика који прате наставу, а између којих седи Милан. Када се низ свих ученика који прате сортира (и када му се додају нова два члана на крајевима реда) ученици који ће сазнати тајну су они који седе између два узастопна члана низа. Ова два члана низа се могу наћи грубом силом.

```
n=int(input())
m=int(input())
prate=[0]
a=input().split()
for i in range(m):
    prate.append(int(a[i]))
prate.append(n+1)
prate=sorted(prate)
r=int(input())
while r>0:
    milan=int(input())
    veci=0
    while prate[veci]<milan:
        veci+=1
    print(prate[veci]-prate[veci-1]-1)
    r-=1
```

Опис главног решења

Решење које ће да ради за све тест примере је веома слично решењу другог подзадатка. Једина разлика је у томе што се узастопни чланови у низу ученика који слушају наставу не тражи грубом силом, већ бинарном претрагом. Потребно је наћи највећи члан низа који седи на месту мањем од Милановог. Следећи члан низа ће сигурно да има веће место од Милановог, па је потребно одузети та два члана и одузети још један.

```
n=int(input())
m=int(input())
prate=[0]
a=input().split()
for i in range(m):
    prate.append(int(a[i]))
prate.append(n+1)
prate=sorted(prate)
r=int(input())
while r>0:
    milan=int(input())
    l=0
    d=m+2
    while l<d-1:
        s=(l+d)//2
        if prate[s]<=milan:
            l=s
        else:
            d=s
    print(prate[l+1]-prate[l]-1)
    r-=1
```

Задатак: Хари Потер

Аутор: Душан Попадић

У магијском свету Харија Потера се за плаћање користе галеони, сикли и кнути. Један галеон има 17 сикла, а један сикл 29 кнута. Рон има код себе n кнута и жели да их укрупни што је више могуће (тако да има што је више могуће галеона, па од остатка што је више могуће сикла и што мање преосталих кнута). Колико ће Рон имати галеона, сикла и кнута након укрупњавања?

Опис улаза

У јединном реду стандардног улаза се налази број n $(0 \leq n \leq 5000)$.

Опис излаза

У јединном реду стандардног излаза исписати три броја - колико ће Рон имати галеона, сикла и кнута након укрупњавања.

Пример 1

Улаз	*Излаз*	*Објашњење*
`1249`	`2 9 2`	Рон жели да има што више галеона, што је у овом случају 2. Када кнуте претвори у 2 галеона, од остатка жели да што више претвори у сикле и добија 9 сикла. Остаје му 2 кнута. Провера решења: два галеона је 986 кнута, 9 сикла је 261 кнут. Када се сабере 986+231+2 добије се 1249 што је почетни број кнута.

Пример 2

Улаз	*Излаз*
`65`	`0 2 7`

Решење

Опис главног решења

Потребно је кнуте укрупњавати редом, почевши од галеона. Најпре се израчуна максималан број галеона као $g = \lfloor \frac{n}{17 \cdot 29} \rfloor$, а затим се преостали кнути рачунају као остатак при том дељењу. Од преосталих кнута се на исти начин одређује максималан број сикла дељењем са 29, док остатак након тога представља број преосталих кнута. На крају се исписују бројеви галеона, сикла и кнута.

```
n = int(input())

g = n // (17*29)
n = n % (17*29)
s = n // 29
n = n % 29
k = n

print(g, s, k)
```

Задатак: Квазинаучник

Аутор: Милан Вугделија

Ћира веома жели да постане научник. Да би стекао жељено звање и признање, смислио је оригиналну теорију и експеримент којим ће ту теорију да потврди. Експеримент је такав, да се при сваком његовом извођењу као резултат увек добија један од бројева $1, 2, 3, \ldots, n$. Према Ћириној теорији, када се експреримент изведе велики број пута, резултати 1 и n треба да се добију приближно исти број пута. Резултати 2 и $n - 1$ такође треба да се појаве приближно исти број пута, као и резултати 3 и $n - 2$, итд.

Нажалост, Ћирини резултати се прилично разликују од оних које предвиђа његова теорија. Уместо да размисли о разлозима неслагања (што би урадио прави научник), Ћира је одлучио да буде непоштен и да нека извођења експеримента не пријави, односно да у извештају за неке резултате пријави мањи број појављивања од оног који је стварно добио. Ћира је сакрио најмањи могућ број претходно изведених експеримената, тако да добије идеално слагање броја појављивања резултата 1 и n, резултата 2 и $n - 1$ итд. Другим речима, пријављени низ броја појављивања сваког резултата чита се исто слева надесно и здесна налево.

Написати програм који за дате бројеве појављивања резултата које је Ћира добио извођењем експеримента израчунава укупан број извођења експреримента које је Ћира прећутао, као и бројеве пријављених појављивања појединих резултата.

Опис улаза

У првом реду стандардног улаза је цео број n ($1 \leq n \leq 1000$), највећи број који може да се добије као резултат приликом извођења експеримента. У другом реду је n неозначених целих бројева, мањих од 1000, раздвојених по једним размаком, при чему i-ти од тих бројева представља број извођења експеримента у којима је добијен резултат i ($1 \leq i \leq n$).

Опис излаза

У први ред стандардног излаза исписати један цео број, укупан број прећутаних извођења експеримента. У други ред исписати n целих бројева раздвојених по једним размаком, при чему i-ти од тих бројева треба да буде пријављени број извођења експеримента у којима је добијен резултат i.

Пример 1

Улаз	*Излаз*	*Објашњење*
`3` `5 2 7`	`2` `5 2 5`	Из улазних података видимо да се у овом примеру као резултат извођења Ћириног експеримента увек добија резултат 1, 2 или 3, као и да је Ћира добио резултат 1 пет пута, резултат 2 два пута, а резултат 3 седам пута. Да би добио резултат који се идеално слаже са његовом теоријом, Ћира је прећутао 2 извођења експеримента при којима је добио резултат 3. Према томе, Ћира је пријавио да се резултат 1 појавио пет пута, резултат 2 два пута, а резултат 3 пет пута.

Пример 2

Улаз	*Излаз*	*Објашњење*
`6` `4 7 5 6 8 2`	`4` `2 7 5 5 7 2`	Ћира је прећутао укупно 4 извођења експеримента, од тога два извођења са резултатом 1 (од 4 извођења пријавио је само 2), једно са резултатом 4 (од 6 извођења пријавио је 5) и једно са резултатим 5 (од 8 извођења пријавио је 7).

Решење

Опис главног решења

Нека се резултат 1 појавио P_1 пута, а резултат n P_n пута. Пошто је Ћира сакрио најмањи могућ број претходно изведених експеримената тако да добије идеално слагање, закључујемо следеће:

- ако је $P_1 < P_n$, Ћира је сакрио $P_n - P_1$ извођења са резултатом n,
- ако је $P_1 > P_n$, Ћира је сакрио $P_1 - P_n$ извођења са резултатом 1,
- ако је $P_1 = P_n$, Ћира од ових извођења није сакрио ништа.

Исто важи за све остале парове резултата, за које је збир резултата једнак $n + 1$.

Према томе, за сваки пар резултата L, D, таквих да је $L + D = n + 1$, треба бројати (додати на бројачку променљиву) $|P_L - P_D|$ сакривених резултата и већу од вредности P_L, P_D из учитаног низа променити тако да буде једнака мањој.

На крају треба приказати број прећутаних извођења експеримента и измењени низ. Наравно, у програму су сви индекси умањени за 1, јер је индекс почетног елемента 0, а не 1.

```python
n = int(input())
a = list(map(int, input().split()))

levo, desno = 0, n-1
br_sakrivenih = 0
while levo < desno:
    br_sakrivenih += abs(a[levo] - a[desno])
```

```
    if a[levo] < a[desno]:
        a[desno] = a[levo]
    else:
        a[levo] = a[desno]

    levo += 1
    desno -= 1

print(br_sakrivenih)
print(*a)
```

Задатак: Да ли постоји правоугаоник

Да ли постоји правоугаоник са целобројним страницама чија је површина једнака P, а обим једнак O?

Опис улаза

Један ред садржи два цела броја P и O $(1 \leq P \leq 10^{12},\ 2 \leq O \leq 4 \cdot 10^{12})$. Гарантује се да је O паран број.

Додатна ограничења- У 50% тест примера ће важити $1 \leq P \leq 10^6$ и $2 \leq O \leq 4 \cdot 10^6$.

Опис излаза

Ако решење не постоји, исписати `-1`.
Иначе исписати два броја a и b $(a \leq b)$ - димензије правоугаоника.

Пример 1		**Пример 2**	
Улаз	*Излаз*	*Улаз*	*Излаз*
`36 26`	`4 9`	`36 1000`	`-1`

Решење

Опис главног решења

Са стандардног улаза се учитавају два цела броја `P` и `O`, који представљају површину и обим правоугаоника. Тражимо да ли постоје целобројне димензије правоугаоника `a` и `b` такве да важи:

- $a \cdot b = P$;
- $2 \cdot (a + b) = O$.

Пошто је гарантовано да је O паран број, можемо да поделимо једначину за обим са 2 и добијамо услов $a + b = O/2$.

Зато у програму прво израчунавамо `zbir = O / 2`. Сада проблем постаје: да ли постоји пар делилаца броја `P` чији је збир једнак `zbir`.

Да бисмо то проверили, пролазимо кроз све позитивне делиоце `delilac` броја `P` до `sqrt(P)`. За сваки такав `delilac` који дели `P`, добијамо један кандидат-пар димензија:

- `a = delilac`;
- `b = P / delilac`.

Затим проверавамо да ли важи `a + b == zbir`. Ако важи, нашли смо правоугаоник који има тражену површину и обим. Пошто треба исписати `a <= b`, по потреби заменимо вредности, сачувамо решење и прекидамо претрагу (јер је довољно наћи било које важеће решење).

Ако након испитивања свих делилаца не пронађемо ниједан пар који задовољава услов, онда решење не постоји и исписујемо `-1`. У супротном, исписујемо пронађене димензије `a` и `b`.

Сложеност алгоритма је $\mathcal{O}(\sqrt{P})$, јер се пролази кроз све делиоце до корена броја P. Просторна сложеност је $\mathcal{O}(1)$, пошто се користи само константан број помоћних променљивих.

Задатак: Сума свих поднизова

Аутор: Александар Николић

Дат је низ целих бројева $a_1, a_2, \dots, a_n$. Израчунати збир свих бројева из свих поднизова овог низа.

Подниз низа је сваки низ који се може добити брисањем неколико (могуће нула) елемената са почетка и неколико (могуће нула) елемената са краја датог низа.

На пример, за низ $[1, 2, 3]$ поднизови су: [1], [2], [3], $[1, 2]$, $[2, 3]$, $[1, 2, 3]$. Збир свих њихових елемената је $1 + 2 + 3 + 3 + 5 + 6 = 20$.

Опис улаза

Први ред стандардног улаза садржи цео број n - број чланова низа.

Други ред стандардног улаза садржи n размаком раздвојених целих бројева $a_1, a_2, \dots, a_n$ - чланови низа.

Опис излаза

Исписати један цео број - збир свих бројева из свих поднизова датог низа.

Ограничења

- $1 \leq n \leq 200.000$
- $-100 \leq a_i \leq 100$

Тест примери су подељени у три групе:

- у тест примерима вредним 20 поена важи: $n \leq 100$;
- у тест примерима вредним 30 поена важи: $n \leq 1000$;
- у тест примерима вредним 50 поена нема додатних ограничења.

Пример

Улаз	*Излаз*
`3` `1 2 3`	`20`

Решење

Опис главног решења

Дат је низ целих бројева `a` дужине `n`. Потребно је израчунати збир свих елемената свих поднизова датог низа. Директно набрајање свих поднизова није изводљиво, јер их има $\mathcal{O}(n^2)$, па би такво решење било преспоро за велика ограничења.

Зато посматрамо допринос сваког појединачног елемента укупном збиру. Фиксирамо позицију `i` и елемент `a[i]`. Питање је у колико поднизова се овај елемент појављује.

Подниз је одређен избором почетне и крајње позиције. Да би `a[i]` био садржан у поднизу, почетак подниза може бити било која позиција од `1` до `i`, а крај подниза било која позиција од `i` до `n`. Зато број поднизова који садрже елемент `a[i]` износи: - i могућих избора за почетак, - $n - i + 1$ могућих избора за крај.

Укупно, елемент `a[i]` се појављује у $i \cdot (n - i + 1)$ поднизова, па његов укупни допринос збиру износи: $i \cdot (n - i + 1) \cdot a[i]$.

Сабирањем овог доприноса за све позиције $i = 1, 2, \dots, n$ добијамо тражени збир свих елемената свих поднизова.

Временска сложеност алгоритма је $\mathcal{O}(n)$, јер се низ обилази једном. Просторна сложеност је $\mathcal{O}(n)$ за чување низа (или $\mathcal{O}(1)$ додатног простора ако се елементи обрађују током читања).

Задатак: Слагалица

Аутор: Огњен Тешић

У току је популарни телевизијски квиз *Слагалица*, тачније игра *Ко зна зна*, у којој такмичари добијају питања једно за другим. Правила су следећа: - тачан одговор доноси **10 поена**; - нетачан одговор одузима **5 поена**; - ако такмичар не одговори на постављено питање, број поена му остаје **непромењен**.

После досадашњег тока игре, резултат је: - Први играч има a поена; - Други играч има b поена; - важи да први играч води, тј. $a > b$.

Питања се настављају. Одредити **најмањи број наредних питања** који морају да се одиграју да би Други играч могао да има **(строго) више поена** од Првог играча, уз најповољнији могући распоред тачних и нетачних одговора за оба такмичара.

Опис улаза

Једина линија улаза садржи два природна броја a и b - тренутни број поена Првог и Другог играча. Гарантовано је да важи $a > b$.

Опис излаза

Исписати један број - најмањи број наредних питања после којих Други играч може да има више поена од Првог.

Пример 1

Улаз	*Излаз*	*Објашњење*
`14 7`	`1`	*Објашњење:* Други играч тачно одговори (+10), Први промаши или не одговори.

Пример 2

Улаз	*Излаз*	*Објашњење*
`50 0`	`4`	*Објашњење:* Разлика је 50. Једно питање може да промени однос за највише 15 поена ($+10$ за Другог и -5 за Првог). Да би се надокнадило 50 поена, потребно је најмање 4 питања.

Пример 3

Улаз	*Излаз*
`62 2`	`5`

Решење

Опис главног решења

Најпре посматрамо разлику у поенима између играча, коју рачунамо као $d = a - b$.

Циљ је да одредимо најмањи број наредних питања након којих Други играч може да има **строго више поена** од Првог.

У једном питању, у најповољнијем случају по Другог играча, може да се деси следеће: - Други играч тачно одговори и **добије +10 поена**; - Први играч нетачно одговори и **изгуби 5 поена**.

У том случају, разлика у поенима се смањује за укупно **15 поена**. Очигледно је да се у једном питању не може надокнадити више од 15 поена разлике.

Зато тражимо најмањи број питања k такав да након k питања важи $15 \cdot k > d$.

Овај услов решавамо тако што израчунамо горњи цео део количника $\frac{d}{15}$, односно: $k = \lceil \frac{d}{15} \rceil$.

У програму се то постиже формулом $k = \frac{d+15}{15}$, уз коришћење целобројног дељења.

Добијена вредност k представља најмањи број наредних питања након којих Други играч може да има више поена од Првог и она се исписује као коначан резултат.

Задатак: Гумене бомбоне

Аутор: Милан Коцић

Отац је купио три кесе гумених бомбона, а тежина сваке кесе је позната. Има два детета и жели да свако дете добије по две кесе тако да укупна тежина код оба детета буде иста. Да би то омогућио, купује још једну кесу и жели да потроши најмање могуће новца. Одредити минималну тежину четврте кесе која обезбеђује да се кесе могу распоредити у два пара једнаке укупне тежине.

Опис улаза

У првом реду стандардног улаза се налазе три природна броја (сваки између 130 и 250) који представљају тежине три кеса бомбона.

Опис излаза

На стандардни излаз исписати један природан број - тежину четврте кесе бомбона.

Пример 1

Улаз	*Излаз*
`146 221 204`	`129`

Пример 2

Улаз	*Излаз*
`200 200 200`	`200`

Решење

Опис главног решења

Најпре уочавамо да у једном пару мора да се нађе најтежа постојећа кеса, јер ће она иначе одредити већу укупну тежину. Да би оба пара имала исту тежину, збир преостале две постојеће кесе и четврте кесе мора бити једнак тежини пара који садржи најтежу кесу два пута. Зато је минимална тежина четврте кесе једнака разлици између збира све три кесе и двоструке тежине најтеже кесе.

Задатак: Адвокатица

Софија је једна од најбољих адвокатица у Београду. Због тога много људи жели да буду њени клијенти. Сваког дана одређени број људи затражи од Софије да их она заступа. Ипак, због тога што је њен посао захтеван, она у једном тренутку може да заступа највише K клијената, те прихвата сваког дана онолико захтева колико може. Софија зна да предмет сваког клијента може да реши за тачно D дана. Она жели да зна колико ће клијената заступати у наредних N дана како би могла да испланира свој дуго очекивани одмор.

Опис улаза

У првом реду стандардног улаза налазе се природни бројеви K, D и N $(1 \leq K \leq 10, 1 \leq D \leq N \leq 100)$, редом највећи број клијената које у једном тренутку Софија може да заступа, број дана потребан за решавање предмета једног клијента и број дана за које Софију занима колико ће клијената заступати.

У другом реду стандардног улаза налази се N природних бројева X_i $(1 \leq X_i \leq 100)$, при чему број X_i представља број људи који су затражили да их Софија заступа.

Опис излаза

У једином реду стандардног излаза исписати један број који представља колико клијената ће Софија заступати док не оде на одмор.

Пример 1

Улаз	*Излаз*	*Објашњење*
`3 2 7` `1 0 1 2 3 0 4`	`8`	Софија прихвата једног клијента првог дана и тај предмет завршава до краја другог дана. Трећег дана прихвата још једног клијента чији предмет завршава до краја четвртог дана. Четвртог дана прихвата још два клијента и њихове предмете завршава до краја шестог дана. Петог дана Софија може да прихвати само једног клијента, јер у том тренутку већ има два клијента која је прихватила четвртог дана. Најзад, Софија прихвата три клијента седмог дана. Укупно Софија заступа 8 клијената.

Пример 2

Улаз	*Излаз*
`3 3 10` `4 3 5 2 6 3 9 8 1 2`	`11`

Решење

Опис главног решења

Приметимо да ако Софија неког дана активно ради на T предмета, она ће тог дана преузети $\min\{K - T, X_i\}$ нових предмета. С једне стране, ако је $K - T$, односно број који представља колико нових предмета би Софија могла да преузме, мањи од X_i, она ће тог дана преузети нових $K - T$ предмета. Са друге стране, ако јој се тог дана јави мање нових клијената него што би она могла да преузме предмета (дакле, $X_i < K - T$), у том случају она преузима нових X_i предмета.

Приметимо и да ако Софија i-тог дана преузме Y_i нових предмета, дана $i + D$ треба умањити њен број активних предмета за Y_i пре него што се започне са одређивањем колико нових предмета ће она преузети тог дана.

Задатак: Слатки поднизови

Дат је низ целих бројева a дужине n. Рећи ћемо да је низ *сладак* ако се може поделити на више узастопних делова (блокова), при чему сваки блок почиње бројем који означава његову дужину, а одмах након њега следи тачно толико елемената тог блока.

На пример,

- $[4,\ \underline{2,\ 3,\ 5,\ 2},\ 1,\ \underline{6}]$ је сладак, јер прво стоји број 4 па затим 4 елемента блока, а после њега блок дужине 1 са једним елементом;
- $[2,\ \underline{7,\ 4},\ 4,\ \underline{2025,\ 2,\ 6,\ 1}]$ је такође сладак.

С друге стране, низови попут $[2]$, $[1,\ 5,\ 3]$ или $[3,\ 12,\ 1]$ немају тражену структуру и зато нису слатки.

У једном потезу дозвољено је из низа обрисати било који елемент. Потребно је одредити најмањи број брисања којима се дати низ може довести да буде сладак.

Опис улаза

Прва линија садржи један цео број n ($1 \leq n \leq 2 \cdot 10^5$) - дужину низа a.

Друга линија садржи n целих бројева $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^6$) - елементе низа a.

*Додатна ограничења*Тест примери су подељени у четири групе: - У тест примерима вредним 25 поена важи $n \leq 15$; - У тест примерима вредним 10 поена су сви чланови низа јединице; - У тест примерима вредним 25 поена важи $n \leq 1000$; - У тест примерима вредним 40 поена нема додатних ограничења.

Опис излаза

Исписати један цео број - минималан број брисања потребан да дати низ постане сладак.

Пример 1

Улаз	*Излаз*	*Објашњење*
`5` `1 2 3 4 5`	`2`	*Објашњење.* Ако се обришу први и последњи елемент, остаје $[2, \underline{3, 4}]$, што одговара блоку дужине 2, па је низ сладак.

Пример 2

Улаз	*Излаз*	*Објашњење*
`8` `2 7 4 4 2025 2 6 1`	`0`	*Објашњење.* Дати низ се налази у поставци и већ је сладак, па није потребна ниједна операција.

Пример 3

Улаз	*Излаз*	*Објашњење*
`5` `1 2 3 1 2`	`1`	*Објашњење.* Дати низ није сладак, а брисањем средњег члана остаје $[1, \underline{2}, 1, \underline{2}]$, те је одговор 1.

Решење

Опис главног решења

Посматрамо низ `arr` дужине `n`. Желимо да обришемо најмањи број елемената тако да преостали елементи (у истом редоследу) формирају *сладак* низ, тј. да се могу поделити на блокове облика:

- прво стоји број `L` (дужина блока),
- одмах затим следи тачно `L` елемената тог блока.

Задатак решавамо динамичким програмирањем над позицијама.

Нека је `dp[i]` минималан број брисања који је потребан да бисмо од подниза који почиње на позицији `i` (од `arr[i]` до краја) могли да добијемо сладак низ.

Тада на позицији `i` имамо две природне могућности:

1. **Обришемо елемент `arr[i]`.**
   Тада прелазимо на позицију `i+1`, а број брисања се увећава за 1, па је
   `a = dp[i+1] + 1`.
2. **Задржимо `arr[i]` као дужину првог блока.**
   Ако је `arr[i] = L`, онда блок мора да заузме тачно `1 + L` елемената: један је сама дужина, а после њега иде `L` елемената блока.
   То значи да после тог блока следећи блок (ако постоји) почиње на позицији
   `i + L + 1`.

   Ако можемо да „скочимо” на ту позицију, онда је цена ове опције
   `b = dp[i + L + 1]`.

   Међутим:

   - ако је `i + L + 1 == n`, онда смо тачно стигли до краја и цена је `0` (све се лепо завршило на крају низа);
   - ако је `i + L + 1 > n`, онда не можемо формирати блок те дужине јер нема довољно елемената, па ову опцију проглашавамо невалидном.

Да бисмо ово једноставно имплементирали, функција `getDP(pos, n, dp)` враћа: - `0` ако је `pos == n` (тачно крај), - невалидну велику вредност `n+1` ако је `pos > n`, - иначе `dp[pos]`.

Зато за свако `i` важи прелаз:
`dp[i] = min(dp[i+1] + 1, getDP(i + arr[i] + 1, n, dp))`.

Базни случајеви: - Ако смо на крају (`i == n`), нема више шта да се поправља: `dp[n] = 0`. - За `i = n-1` (један елемент до краја), једини начин да буде сладак је да га обришемо, па је `dp[n-1] = 1`.

DP попуњавамо уназад (од `n-2` до `0`), јер `dp[i]` зависи од већ израчунатих вредности `dp[i+1]` и `dp[i + arr[i] + 1]`.

Коначан одговор је `dp[0]`, јер он представља минималан број брисања потребан да цео низ постане сладак.

Временска сложеност алгоритма је $\mathcal{O}(n)$, јер се свака позиција низа обрађује тачно једном, а сваки прелаз се извршава у константном времену. Просторна сложеност је $\mathcal{O}(n)$, због коришћења низа `dp` дужине `n+1`.

Глава 3

Општинско такмичење

Задатак: Буџет за летовање

Аутори: Ђура Пађан, Филип Марић

Породица Јовановић планира трошкове свог летовања. За превоз ће платити износ од P динара у сваком смеру. На летовању ће остати n дана. Смештај и храну ће сваки дан плаћати по S динара. Напиши програм који одређује колико новца ће породица Јовановић потрошити на летовању.

Опис улаза

Са стандардног улаза се уноси природан број P ($1000 \leq P \leq 20000$), затим број дана n ($3 \leq n \leq 15$) и природан број S ($1000 \leq S \leq 20000$). Сваки број се уноси у посебном реду.

Опис излаза

На стадардни излаз исписати укупне трошкове летовања.

Пример

Улаз	*Излаз*	*Објашњење*

`5000`
`7`
`3000`
	`31000`	Укупни трошкови су трошкови пута (10000) и трошкови боравка (21000) што је укупно 31000.

Решење

Пошто се плаћа по P данара за сваки смер путовања, укупни трошкови превоза су $2 \cdot P$. За смештај и храну ће платити $n \cdot S$. Стога ће укупни трошкови бити $2 \cdot P + n \cdot S$.

```
P = int(input())
n = int(input())
S = int(input())
print(2*P + n*S)
```

Задатак: Упоредити цифре

Аутор: Огњен Тешић

Дат је један четвороцифрен број.

Средње цифре четвороцифреног броја су **цифра стотина** и **цифра десетица**.
Потребно је упоредити те две цифре.

- ако је цифра стотина већа од цифре десетица, исписати знак >
- ако је мања, исписати знак <
- ако су једнаке, исписати знак =

Опис улаза

Са стандардног улаза се учитава један четвороцифрен број.

Опис излаза

На стандардни излаз исписати један од знакова
>, < или = у зависности од поређења средњих цифара.

Пример 1

Улаз	*Излаз*	*Објашњење*
`5834`	`>`	Цифра стотина је 8, а цифра десетица је 3. Пошто је 8 веће од 3 исписујемо `>`.

Пример 2

Улаз	*Излаз*
`2447`	`=`

Пример 3

Улаз	*Излаз*
`3192`	`<`

Решење

Главно питање у овом задатку је како из четвороцифреног броја издвојити цифре десетица и стотина. Да бисмо то урадили потребно је прво приметити да када број поделимо са 10 (користећи целобројно дељење) практично бришемо његову последњу цифру (на пример 2645 постаје 264), а када одредимо остатак при дељењу са 10 добијамо последњу цифру (2645 даје остатак 5 при дељењу са 10).

Да бисмо издвојили цифру десетица (другу са десна) потребно је да прво “обришемо” последњу цифру дељењем са 10 и онда издвојимо нову последњу цифру одређивањем остатка при дељењу са 10. Дакле цифру десетица добијамо као $(n//10)$.

Да бисмо издвојили цифру стотина (трећу са десна) потребно је да прво “обришемо” последње две цифре тако што два пута поделимо са 10 (тј. једном са 100) и онда издвојимо нову последњу цифру одређивањем остатка при дељењу са 10. Дакле цифру десетица добијамо као $(n//100)$. .

```python
n = int(input())

stotine = (n // 100) % 10
desetice = (n // 10) % 10

if stotine > desetice:
    print(">")
elif stotine < desetice:
    print("<")
else:
    print("=")
```

Задатак: Дељивост са 3 7 21

- Аутори: Ђура Пађан, Филип Марић *

Број 14 је дељив са 7, али не и са 3, број 15 је дељив са 3, али не и са 7, док је број 42 дељив и са 3 и са 7 (па је зато дељив и са 21). Напиши програм који испитује дељивост унетог броја са 3, 7 и 21.

Опис улаза

Са стандардног улаза се уноси природан број n ($1 \leq n \leq 1\,000\,000$).

Опис излаза

На стандардни излаз исписати:

- `3` ако је број дељив са 3, али не и са 7.
- `7` ако је број дељив са 7, али не и са 3.
- `21` ако је број дељив са 21,
- `-` ако број није дељив ни са 3 ни са 7.

Пример 1		**Пример 2**		**Пример 3**		**Пример 4**	
Улаз	*Излаз*	*Улаз*	*Излаз*	*Улаз*	*Излаз*	*Улаз*	*Излаз*
`14`	`7`	`15`	`3`	`16`	`-`	`63`	`21`

Решење

Прво можемо проверити да ли је број дељив са 21 (израчунавањем остатка при дељењу и провером да ли је тај остатак 0). Ако јесте, исписујемо `21`. Ако није дељив са 21, проверавамо да ли је дељив са 7. Ако јесте, исписујемо `7`. У супротном проверавамо да ли је дељив са 3. Ако јесте, исписујемо `3`. У супротном знамо да број није дељив ни са 7 ни са 3, па исписујемо `-`.

Задатак: Распоређивање слика

Аутори: Филип Марић, Душан Попадић

На интернет страници слике се слажу једна поред друге у ред. Слике могу бити различите висине и ширине. Слике се равнају тако да им доње ивице буду у линији, а десна ивица тренутне слике се поклапа са левом ивицом наредне слике. На сајт треба сложити 4 слике. Потребно је одредити колика ће бити висина и ширина реда слика уколико су познате димензије сваке појединачно.

Опис улаза

У 4 реда стандардног улаза се уносе по два броја који представљају редом ширину и висину слике. Ови бројеви су природни и највише 100.

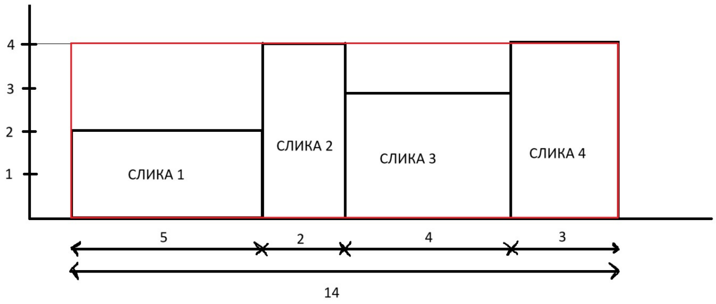
Опис излаза

У једном реду стандардног излаза исписати два природна броја: прво ширину, а затим висину реда који садржи све слике.

Пример 1

Улаз	*Излаз*	*Објашњење*

`5 2`
`2 4`
`4 3`
`3 4`
	`14 4`	Када се слике ређају једна поред друге ширина реда у који се ређају ће бити 14. Висина реда је таква да у њега може да стане и највиша слика - дакле 4. Погледај слику која иде уз пример. Црвеним правоугаоником је обележен ред на страници који слике заузимају.



Пример 2

Улаз	*Излаз*

`2 6`
`17 5`
`12 6`
`14 2`
	`45 6`

Пример 3

Улаз	*Излаз*

`18 3`
`15 3`
`1 3`
`1 3`
	`35 3`

Решење

Пошто се слике ређају у ред једна поред друге, укупна ширина реда биће једнака збиру ширина свих слика. Слике могу имати различите висине, а висина реда је одређена висином највише слике. Дакле, да би се одредила ширина реда потребно је сабрати ширине свих слика, а да би се одредила висина потребно је одредити максимум висина свих слика.

Задатак: Свеска

Аутор: Огњен Тешић

Сара је купила свеску и сваког месеца попуњавала исти број страна (цео број). После T месеци, у свесци је било попуњено укупно X страна.

Ако данас у свесци има укупно Z попуњених страна, одредити колико је месеци прошло од тренутка када је Сара почела да користи свеску.

Приметите да ће тражени одговор бити цео број.

Опис улаза

У три реда налазе се три цела броја: - у првом реду број T, - у другом реду број X, - у трећем реду број Z.

Опис излаза

Исписати један цео број - број месеци који је прошао од почетка коришћења свеске.

Ограничења

- $1 \leq T \leq 1000$
- $1 \leq X \leq 1000$
- $1 \leq Z \leq 1000$

Пример 1

Улаз	*Излаз*	*Објашњење*

`4`
`20`
`45`
	`9`	*Објашњење.* После 4 месеца било је попуњено 20 страна, што значи да је Сара попуњавала 5 страна месечно. Ако је данас попуњено 45 страна, прошло је $45/5 = 9$ месеци.

Пример 2

Улаз	*Излаз*

`6`
`30`
`55`
	`11`

Решење

Ако је после T месеци Сара попунила X страна свеске то значи да је попуњавала $po\_mesecu = X/T$ страна сваког месеца. Приметимо да је у тексту задатка (у првој реченици) наглашено да ће овај количник сигурно бити цео број. Ако са $meseci$ обележимо колико је прошло месеци од тренутка када је Сара почела да попуњава свеску до данас, тада је број попуњених страна до данас једнак $Z = po\_mesecu \cdot meseci$, одакле добијамо да је $meseci = Z/po\_mesecu$, а то је вредност која нам се тражила у задатку.

```
T = int(input())
X = int(input())
Z = int(input())

po_mesecu = X // T
meseci = Z // po_mesecu

print(meseci)
```

Задатак: Верзије софтвера

Аутор: Филип Марић

Свака верзија софтвера се означава са 3 броја. На пример 2.13.5 је пета подподверзјиа тринаесте подверзије друге верзије софтвера. Написати програм који проверава да ли тренутно инсталирана верзија софтвера на рачунару задовољава потребе корисника.

Опис улаза

Са стандардног улаза се уноси верзија софтвера коју потребно имати на рачунару и верзија софтвера која је тренутно присутна (по три природна броја раздвојена размацима, свака у посебном реду, прво потребна, па инсталирана).

Опис излаза

На стандардни излаз исписати `da` ако је тренутна верзија софтвера једнака потребној или је новија од тога тј. `ne` у супротном.

Пример 1

Улаз	*Излаз*	*Објашњење*
`2 5 3` `2 6 1`	`da`	Тренутна инсталирана верзија је 2.6.1 што је новија верзија у односу на потребну 2.5.3.

Пример 2

Улаз	*Излаз*
`2 5 3` `2 4 14`	`ne`

Решење

Пошто не знамо колико цифара имају ознаке није једноставно три броја претворити у један. Зато ћемо верзије поредити лексикографски. Прво се пореди први број, па ако је он једнак пореди се други број, па ако је и он једнак, пореди се трећи број.

Можемо написати један логички израз којим се проверава да ли је инсталирана верзија у реду.

```
potrebnoA, potrebnoB, potrebnoC = map(int, input().split())
instaliranoA, instaliranoB, instaliranoC = map(int, input().split())

if (instaliranoA > potrebnoA or
    (instaliranoA == potrebnoA and instaliranoB > potrebnoB) or
    (instaliranoA == potrebnoA and instaliranoB == potrebnoB and instaliranoC > potrebnoC))
    print("da")
else:
    print("ne")
```

Уместо јединственог логичког израза можемо надовезати испитивање више једноставних услова.

```
potrebnoA, potrebnoB, potrebnoC = map(int, input().split())
instaliranoA, instaliranoB, instaliranoC = map(int, input().split())

if instaliranoA > potrebnoA:
    OK = True
elif instaliranoA < potrebnoA:
    OK = False
elif instaliranoB > potrebnoB:
    OK = True
elif instaliranoB < potrebnoB:
    OK = False
elif instaliranoC >= potrebnoC:
    OK = True
else:
    OK = False

if OK:
    print("da")
else:
    print("ne")
```

Алтернативно можемо користити уграђено поређење торки у Пајтону, што нам даје и најједноставније решење.

```
potrebnoA, potrebnoB, potrebnoC = map(int, input().split())
instaliranoA, instaliranoB, instaliranoC = map(int, input().split())

if (potrebnoA, potrebnoB, potrebnoC) <= (instaliranoA, instaliranoB, instaliranoC):
    print("da")
else:
    print("ne")
```

Задатак: Жућков рејон

Аутори: Милан Вугделија, Огњен Тешић

Жућко воли да трчкара дуж улице лево и десно од свог дворишта, њушкајући земљу између трчкарања.

Написати програм који за дато Жућково кретање одређује колико се он најдаље кретао лево и десно од дворишта.

Опис улаза

У првом реду стандардног улаза је број Жућкових трчкарања, цео позитиван број не већи од 100. У другом реду је n целих бројева a_i различитих од 0, од којих сваки преставља дужину једног Жућковог трчкарања. Вредност $a_i > 0$ означава да је Жућко претрчао a_i метара надесно, док вредност $a_i < 0$ означава да је Жућко претрчао $|a_i|$ метара налево. Бројеви a_i су по апсолутној вредности највише 200.

Опис излаза

У први ред стандардног излаза исписати колико највише метара је жућко ишао лево од капије свог дворишта, а у други ред колико је највише ишао десно.

Пример

Улаз	*Излаз*	*Објашњење*
`5` `3 -5 2 6 -4`	`2` `6`	Откако је изашао из свог дворишта, Жућко је ишао 3 метра надесно, затим 5 метара налево (то је до места које је 2 метра лево од капије), затим 2 метра надесно (до своје капије), па 6 метара надесно (и толико десно од капије) и на крају 4 метра налево. Према томе, Жућко је ишао највише 2 метра лево и 6 метара десно од своје капије.

Решење

Решење одређује интервал кретања Жућка праћењем његове позиције након сваког трчкарања. Програм иницијално поставља позицију на 0 и затим пролази кроз сваки померај, ажурирајући тренутну позицију. Истовремено бележи најудаљеније тачке лево и десно од полазне позиције. На крају израчунава и приказује апсолутну вредност најудаљеније леве тачке и вредност најудаљеније десне тачке.

```
n = int(input())
trk = [int(x) for x in input().split()]
polozaj, krajnji_levi, krajnji_desni = 0, 0, 0
for i in range(n):
    pomak = trk[i]
```

```
    polozaj += pomak
    krajnji_levi = min(krajnji_levi, polozaj)
    krajnji_desni = max(krajnji_desni, polozaj)

print(-krajnji_levi)
print(krajnji_desni)
```

Задатак: Јелка

Аутори: Филип Марић, Михајло Марковић Како је Нова година била пре тачно месец дана, Мали Перица је решио да раскити јелку и спакује украсе како би их сачувао за следећу годину. Међутим, пре него што почне, жели да преброји колико украса има, како би одабрао одговарајућу кутију у коју ће их упаковати.

Пошто је јелка веома велика, Мали Перица је одлучио да је обради рачунарски.
Слика јелке је представљена помоћу N **ниски**, при чему: - `.` означава део слике на коме нема јелке - `*` означава део јелке који није украшен - `o` означава украс на јелци

Помозите Малом Перици тако што ћете му на основу датих ниски рећи колико украса се налази на јелци.

Опис улаза

У првом реду уноси се природан број N $(1 \leq N \leq 100)$, број ниски које представљају јелку.

У наредних N редова уносе се ниске, свака дужине $2N-1$. Гарантује се да се свака ниска састоји искључиво од карактера `.`, `*` и `o`, као и да дате ниске заиста формирају слику јелке, тј. да у i-том реду постоји тачно $2i-1$ карактера који нису `.` и сви они су узастопни и налазе се у средини i-те ниске (видети пример).

Опис излаза

У једном реду стандардног излаза исписати један број који представља број украса на јелци.

Пример

Улаз	*Излаз*

`4`
`...*...`
`..*o*..`
`.*o***.`
`o*oo***`
	`5`

Решење

Потребно је у једној променљивој чувати укупан број пронађених украса. За сваку учитану ниску пролазимо кроз сваки њен карактер и проверавамо да ли је једнак ‘o’. Ако јесте, увећавамо резултат за један.

Задатак: Питагорина тројка

Аутори: Ђура Пађан, Огњен Тешић

Три броја чине Питагорину тројку ако важи да је квадрат једног броја једнак збиру квадрата преостала два броја. Испитати да ли дата три броја чине Питагорину тројку.

Опис улаза

У једином реду улаза су дата три природна броја мања од 1000, раздвојена размаком.

Опис излаза

На излаз исписати `da` ако дата три броја чине Питагорину тројку, односно `ne` уколико је не чине.

Пример 1

Улаз	*Излаз*	*Објашњење*
`3 5 4`	`da`	*Објашњење примера.* Важи $5^2 = 3^2+4^2$, па ова три броја чине Питагорину тројку.

Пример 2

Улаз	*Излаз*
`5 4 6`	`ne`

Пример 3

Улаз	*Излаз*	*Објашњење*
`13 12 5`	`da`	*Објашњење примера.* Важи $13^2 = 5^2 + 12^2$, па ова три броја чине Питагорину тројку.

Решење

Решење проверава сва три могућа случаја за Питагорину тројку: да ли је квадрат једног броја једнак збиру квадрата преостала два броја. Проверавамо услове за сваку комбинацију бројева a, b и c. Ако један од услова важи, исписујемо „да”, иначе „не”.

Задатак: Учешће

Аутор: Душан Попадић

На семинару „Млади рођени 90-их” право учешћа имају сви, али приоритет имају они који су рођени 90-их (од 1990. до 1999. године). Уколико су по овом критеријуму две особе једнаке, приоритет има млађа. За последње слободно место су се пријавили Роберт и Лазар. Одредите који ће од њих двојице бити примљен на семинар.

Опис улаза

У првом реду се налази Робертова година рођења. У другом реду се налази Лазарева година рођења.

Опис излаза

Уколико Роберт улази на семинар исписати R, уколко улази Лазар исписати L, а уколико је немогуће одлучити на основу године рођења исписати N.

Пример 1

Улаз	*Излаз*	*Објашњење*
`1991` `2000`	`R`	Пошто је Роберт рођен током 90-их он има предност у односу на Лазара.

Пример 2

Улаз	*Излаз*	*Објашњење*
`1994` `1997`	`L`	Пошто су и Роберт и Лазар рођени током 90-их, предност има Лазар јер је млађи.

Пример 3

Улаз	*Излаз*	*Објашњење*
`1995` `1995`	`N`	Пошто су обојица рођени исте године, није могуће одредити ко има предност само на основу податка о години рођења.

Решење

У овом задатку је потребно проверити неколико случајева:

- Ако су године рођења исте, онда је немогуће одредити ко има предност и потребно је исписати `N`.
- Ако су обе године исте по питању припадности деведесетим (или обе припадају или ниједна не припада) онда треба проверити ко је млађи и исписати његово слово. Обратите пажњу да је млађи онај који има **већу** годину рођења (неко рођен 1997. је млађи од неког рођеног 1991. године).
- Ако претходни услови нису испуњени то значи да једна година припада деведесетим, а друга не и треба исписати слово оног који јесте рођен деведесетих.

Пошто је на неколико места у коду потребно проверити да ли су Роберт и Лазар рођени деведесетих, направили смо помоћну функцију која то за нас проверава како би код био прегледнији.

```python
# odredjuje da li godina pripada devedesetim godinama
def dev(g):
    return 1990 <= g <= 1999

r = int(input())
l = int(input())

# ako su godine jednake, nemoguce je odrediti ko ima prednost
if r == l:
    print("N")
else:
    # ako su isti po kriterijumu pripadanja devedesetim onda se bira mladji
    if dev(r) == dev(l):
        if r > l:
            print("R")
        else:
            print("L")
    # u ove grane ispod se ulazi ako jedan pripada devedesetim, a drugi ne
    elif dev(r):
        print("R")
    else:
        print("L")
```

Задатак: Баундинг бокс

Аутори: Филип Марић, Душан Попадић

На екрану једне апликације нацртан је скуп тачака и потребно је одредити најмањи правоугаоник (чије су странице паралелне координатним осама) који садржи све тачке првог квадранта. Првом квадранту припадају тачке чије су обе координате веће од нуле. Напиши програм који на основу познатих координата тих тачака одређује координате темена тог правоугаоника.

Опис улаза

Са стандардног улаза се учитава број тачака n ($2 \leq n \leq 10^5$), а затим из наредних n редова координате тих тачака (целобројне x и y координате, вредности између -10^5 и 10^5, раздвојене размаком). Сматрати да ће увек потојати барем две тачке у првом квадранту.

Опис излаза

На стандардни излаз исписати координате 4 темена правоугаоника и то доње-лево, доње-десно, горње-лево и горње-десно.

Пример

Улаз	*Излаз*
`3`	`7 20`
`10 30`	`10 20`
`7 20`	`7 30`
`-14 25`	`10 30`

Решење

Да бисмо одредили најмањи правоугаоник који садржи све тачке потребно је да одредимо његове леву, десну, горњу и доњу границу. Да би све тачке биле у правоугаонику мора да важи да су “најлевља”, “најдеснија”, “најгорња” и “најдоња” тачка у правоукаонику. Ако са L_x обележимо X координату најлевље тачке, тада граница лева правоугаоника мора да има X координату која је мања или једнака L_x, а пошто нам је потребан најмањи могући правоугаоник, онда ће лева граница прваоугаоника баш бити једнака L_x. Сличну логику примењујемо и за остале три странице правоугаоника. Дакле, потребно је одредити најмање и највеће X и Y координате које се појављују међу тачкама. Пошто се тражи да се одреди правоугаоник који обухвата тачке само у првом квадранту, приликом учитавања податак ћемо игнорисати све тачке које не припадају првом квадранту.

Задатак: Различите мајице

Аутори: Милан Вугделија, Огњен Тешић

Три сестре желе да носе мајице различите боје. Написати програм који учитава која од њих има које мајице, а исписује број начина да њих три обуку мајице различитих боја.

Опис улаза

Маијце за сваку сестру описане су подацима у два реда стандардног улаза (прва два реда за прву сестру, следећа два за другу, а последња два за трећу). У првом од два реда који садрже податке о мајицама једне сестре налази се број мајица те сестре, а у другом реду боје мајица.

Број мајица по сестри није већи од 10.

Опис излаза

На стандардни излаз исписати један цео број, тражени број начина да сестре обуку мајице различитих боја.

Пример

Улаз

```
3
crvena plava zuta
2
zelena crvena
3
crna zuta crvena
```

Излаз

```
10
```

Објашњење

Могући су следећи избори мајица:

```
crvena zelena crna
crvena zelena zuta
plava zelena crna
plava zelena zuta
plava zelena crvena
plava crvena crna
plava crvena zuta
zuta zelena crna
zuta zelena crvena
zuta crvena crna
```

Решење

Да бисмо пребројали све могуће расподеле мајица потребно је проћи са три угнежђене петље кроз сва три низа. Да бисмо осигурали да ће све три сестре носити различите мајице, при одабиру мајица из другог и трећег низа додајемо услов да се та боја није већ појавила код једне од претходних сестара (ако их има). Када одредимо једну могућу комбинацију боја, увећавамо бројач за 1.

```python
n1 = int(input())
boje1 = [x for x in input().split()]
n2 = int(input())
boje2 = [x for x in input().split()]
n3 = int(input())
boje3 = [x for x in input().split()]

br_nacina = 0
for b1 in boje1:
    for b2 in boje2:
        if b1 == b2:
            continue
        for b3 in boje3:
            if b1 != b3 and b2 != b3:
                br_nacina += 1

print(br_nacina)
```

Глава 4

Окружно такмичење

Задатак: Нз

Аутор: Огњен Тешић

На табли су записана три цела броја. У зависности од унетог слова, потребно је извршити једну од следећих операција над њима:

- ако је унето слово `N`, потребно је одредити најмањи од та три броја;
- ако је унето слово `Z`, потребно је одредити збир два највећа броја.

Опис улаза

У првом реду налази се једно велико слово (`N` или `Z`).
У другом реду налазе се три међусобно различита цела броја размаком раздвојена (апсолутна вредност свих је највише 1000).

Опис излаза

Исписати један цео број - резултат тражене операције.

Пример 1

Улаз	*Излаз*	*Објашњење*
`N` `7 10 3`	`3`	Унето је слово `N`, што значи да треба одредити најмањи од три броја. Најмањи међу њима је 3, па је то тражени резултат.

Пример 2

Улаз	*Излаз*	*Објашњење*
`Z` `4 -2 9`	`13`	Унето је слово `Z`, што значи да треба одредити збир два највећа броја. Највећа два броја су 9 и 4. Њихов збир је 13, па је то тражени резултат.

Решење

Један од начина за решавање овог задатка је да на почетку свакако одредимо најмањи од три броја. Уколико је унето `N` само испишемо тај најмањи. Уколико је унето `Z` исписујемо разлику између збира сва три броја и најмањег броја што је заправо збир два највећа.

```
operacija = input()
a, b, c = map(int, input().split())

najmanji = min(a, b, c)

if operacija == 'N':
    print(najmanji)
else:
    print(a + b + c - najmanji)
```

Задатак: Другарице

Аутор: Теодора Обрадовић

Љубица први пут прави своју пицама журку. Љубица је на журку позвала четири другарице. Међутим, једна од позивница се случајно изгубила у пошти. Због тога су на Љубичину журку дошле само три другарице. Чија позивница се изгубила у пошти?

Опис улаза

У првој линији стандардног улаза уноси се име другарице којој је намењена прва позивница.
У другој линији стандардног улаза уноси се име другарице којој је намењена друга позивница.
У трећој линији стандардног улаза уноси се име другарице којој је намењена трећа позивница.
У четвртој линији стандардног улаза уноси се име другарице којој је намењена четврта позивница.

У петој линији стандардног улаза уноси се име другарице која је прва дошла на журку.
У шестој линији стандардног улаза уноси се име другарице која је друга дошла на журку.
У седмој линији стандардног улаза уноси се име другарице која је последња дошла на журку.

Гарантује се да ће сва четири имена другарица бити различита у свим тест примерима.

Опис излаза

На стандардни излаз исписати име другарице која није добила позивницу.

Пример 1

Улаз	*Излаз*	*Објашњење*

`Julija`
`Ana`
`Lena`
`Nina`
`Lena`
`Julija`
`Nina`
	`Ana`	На основу последње три линије улаза можемо видети да су дошле Лена, Јулија и Нина, а Ана није.

Пример 2

Улаз	*Излаз*

`Mima`
`Gorica`
`Ana`
`Teodorica`
`Ana`
`Gorica`
`Mima`
	`Teodorica`

Решење

За сваку другарицу можемо независно проверити да ли је дошла и ако није исписати њено име. Свако име позване другарице поредимо са сва три имена другарица које су дошле и уколико се испостави да ниједно поређење не врати вредност тачно, то значи да та позвана другарица није дошла и исписујемо њено име.

```python
prvaPozivnica = input()
drugaPozivnica = input()
trecaPozivnica = input()
cetvrtaPozivnica = input()

prviDolazak = input()
drugiDolazak = input()
treciDolazak = input()

if (prvaPozivnica != prviDolazak
    and prvaPozivnica != drugiDolazak
    and prvaPozivnica != treciDolazak):
    print(prvaPozivnica)
elif (drugaPozivnica != prviDolazak
    and drugaPozivnica != drugiDolazak
    and drugaPozivnica != treciDolazak):
    print(drugaPozivnica)
elif (trecaPozivnica != prviDolazak
    and trecaPozivnica != drugiDolazak
    and trecaPozivnica != treciDolazak):
    print(trecaPozivnica)
else:
    print(cetvrtaPozivnica)
```

Задатак: Пикадо

Аутор: Милан Коцић

Марко у својој соби има пикадо, али га мрзи да рачуна бодове, па једино што ради јесте да редом записује резултат сваког од својих n бацања. Игра по поједностављеним правилима: почиње са m бодова, а циљ је да стигне до 0. Ако у неком тренутку има x бодова и погоди поље вредности y, његов нови број бодова постаје $x - y$. Међутим, ако би тим одузимањем резултат постао негативан (односно $x - y < 0$), тај погодак се не рачуна и број бодова остаје исти. Пошто Марко не прати израчунавање током игре, може се десити да баци стрелицу више пута него што је било потребно да заврши игру. Написати програм који ће, на основу његових забележених бацања, одредити да ли је Марко заиста завршио игру, тј. да ли је у неком тренутку достигао резултат 0.

Опис улаза

У првом реду стандардног улаза се налазе два броја m и n, $m \in [1, 10000]$ и $n \in [1, 1000]$. У другом реду се налази n бројева из интервала $[1, 20]$ који представљају резултат сваког бацања.

Опис излаза

На стандардни излаз исписати број бацања после којих је Марко стигао до броја 0. Ако није завршио партију исписати -1.

Пример 1

Улаз	*Излаз*	*Објашњење*
`15 8` `1 2 3 4 7 5 6 8`	`6`	Резултат после сваког бацања је болдован: 15 - 1 = **14** 14 - 2 = **12** 12 - 3 = **9** 9 - 4 = **5** **5** - 7 = -2 (остаје са 5 поена јер не може да има негативан број поена) 5 - 5 = **0**

Пример 2

Улаз	*Излаз*
`4 6` `7 5 6 8 9 15`	`-1`

Решење

Да бисмо решили овај задатак потребно да је симулирамо игру која се дешава. Дакле, посматрамо бацање по бацање и рачунамо број бодова који Марко има након сваког бацања, водећи рачуна да број не сме да падне испод 0. Уколико би неко бацање довело до тога да број бодова падне испод 0, то бацање игноришемо. Симулацију заустављамо или кад потрошимо сва бацања или када Марко достигне тачно 0 бодова. Уколико је Марко у неком тренутку достигао 0 бодова, прекидамо симулацију и исписујемо после колико бацања је то постигао. Уколико прођемо кроз сва бацања и Марко ни у једном тренутку није достигао 0 бодова, исписујемо -1.

```python
m, n = map(int, input().split())

b = 0

bacanja = list(map(int, input().split()))

while b < n and m != 0:

    x = bacanja[b] # trenutno bacanje
    if m - x >= 0: # provera da li je došlo do prebacivanja nule
        m -= x
```

```
    b += 1

if m == 0:
    print(b)
else:
    print(-1)
```

Задатак: Ски стаза

Аутор: Душан Попадић

На недавно завршеним Зимским олимпијским играма у Милану одржане су, између осталог, скијашке трке у супервелеслалому и спусту. Због великих брзина које се остварују у овим дисциплинама, стазе се праве тако да одмах после циља почне блага узбрдица како би такмичари могли да се безбедно зауставе. Дакле, стаза изгледа тако што надморска висина све време опада до циља, а онда креће да расте одмах након циља и све време наставља да расте до краја.

Стаза је описана низом природних бројева који описује како се мења висина скијашке стазе у метрима (први елемент низа представља висину старта). Уносе се подаци о две стазе, за сваки одредити да ли је адекватна за вожњу супервелеслалома и спуста, тј. да ли задовољава услов да висина опада све време до циља, а расте након циља, као и индекс низа на ком се налази висина циља.

Опис улаза

У првом реду налази се број n ($5 \leq n \leq 1000$) - број делова прве стазе. У другом реду налази се n природних бројева (сваки број је између 1700 и 5000) који представљају висине прве стазе редом од старта до краја.

У трећем реду налази се број m ($5 \leq m \leq 1000$) - број делова друге стазе. У четвртом реду налази се m природних бројева (сваки број је између 1700 и 5000) који представљају висине друге стазе редом од старта до краја.

Опис излаза

За сваку стазу, у посебном реду, исписати по један број - индекс низа на ком се налази висина циља. Индекси се броје од 1. Уколико једна од стаза није адекватна, за њу исписати `-1`.

Ограничења

- У тест примерима вредним 30 поена је гарантовано да су обе стазе адекватне.
- У тест примерима вредним 70 поена нема додатних ограничења.

Пример 1

Улаз	*Излаз*

`7`
`2105 2100 2060 2000 1990 1992 1994`
`5`
`3700 3650 3660 3670 3680`
	`5`
	`2`

Објашњење

Висина циља прве стазе је 1990 метара, а број 1990 се налази на 5. месту. Висина циља друге стазе је 3650 метара, а број 3650 се налази на 2. месту.

Пример 2

Улаз	*Излаз*
`9`	`-1`
`2100 2060 2000 1990 1992 1994 1991 1995 2000`	`-1`
`5`	
`2100 2060 2000 1990 1988`	

Објашњење

Прва стаза није адекватна јер од висине 1990 метара почиње да расте, па онда после поново да опада после висине 1994. Друга стаза није адекватна јер нема узбрдице после циља.

Пример 3

Улаз	*Излаз*
`5`	`-1`
`4300 4305 4310 4315 4323`	`-1`
`7`	
`2105 2100 2060 2060 1990 1992 1994`	

Објашњење

Прва стаза није адекватна јер не постоји низбрдица. Друга стаза није адекватна јер има зараван, тј. не опада све време пре циља (два пута заредом је иста висина).

Пример 4

Улаз	*Излаз*
`5`	`-1`
`4500 4501 4499 4502 4498`	`5`
`6`	
`1800 1780 1760 1740 1720 1730`	

Решење

Решење када је гарантовано да су обе стазе адекватне

Можемо приметити да када је стаза адекватна она мора имати облик латиничног слова `V` (не нужно са симетричним крацима) и да је циљ на месту најмање висине стазе. Дакле, за сваки низ је довољно одредити позицију најмањег елемента и исписати је.

Ово решење доноси 30 поена.

```python
def nadjiCilj(s, n):
    minimum = 10000
    min_ind = 0
    for i in range(n):
        si = s[i]
        if si < minimum:
            minimum = si
            min_ind = i
    return min_ind + 1


# ucitavanje
n1 = int(input().strip())
s1 = list(map(int, input().split()))
print(nadjiCilj(s1, n1))
```

```
n2 = int(input().strip())
s2 = list(map(int, input().split()))
print(nadjiCilj(s2, n2))
```

Решење у општем случају

Да бисмо решили задатак у општем случају, поред одређивања позиције минимума, потребно је проверити да ли су стазе адекватне, тј. да ли су облика латиничног слова `V`.

Како бисмо оптимизовали код, логику за проверу да ли је стаза адекватна и за враћање позиције циља стављамо у функцију и позивамо за сваку стазу.

Први услов који мора да буде испуњен је да стаза креће низбрдо и то можемо лако проверити поређењем прва два елемента. Ако је други елемент мањи или једнак првом, знамо да стаза није адекватна и враћамо -1. Након тога помоћу циклуса пролазимо кроз елементе докле год опадају (приметите да ако се у било ком тренутку појаве два елемента исте висине одмах можемо да прекинемо и вратимо -1). Из поменутог циклуса можемо изаћи на два начина:

- елементи почињу да расту што значи да смо нашли циљ
- дошли смо до краја низа, то значи да не постоји део са узбрдицом и можемо вратити -1 јер стаза није адекватна

Чак и ако смо нашли позицију циља и даље нисмо сигурни да је стаза адекватна и морамо проверити да ли се после тог циља налази узбрдица. То проверавамо у циклусу којим се крећемо до краја низа и враћамо -1 уколико се у било ком тренутку деси да низ не расте (дакле елемент је мањи или једнак претходном).

Уколико смо стигли до краја и закључили смо да је стаза адекватна, враћамо позицију циља коју смо одредили на крају првог циклуса.

```
def nadjiCilj(s, n):
    if s[0] <= s[1]:
        return -1

    raste = False
    cilj = -1

    i = 1
    while not raste and i < n:
        if s[i] == s[i - 1]:
            return -1

        if s[i] > s[i - 1]:
            raste = True
        else:
            i += 1

    if i == n:
        return -1

    cilj = i
```

```
    while i < n:
        if s[i] <= s[i - 1]:
            return -1
        i += 1

    return cilj


n1 = int(input())
s1 = list(map(int, input().split()))
print(nadjiCilj(s1, n1))

n2 = int(input())
s2 = list(map(int, input().split()))
print(nadjiCilj(s2, n2))
```

Задатак: Седласти елемент

Аутор: Огњен Тешић

Дата је матрица целих бројева димензија $n \times m$. Елемент матрице a_{ij} називамо *седластим* ако истовремено важи: - a_{ij} је најмањи елемент у свом реду (у i-том реду); - a_{ij} је највећи елемент у својој колони (у j-тој колони).

Написати програм који одређује колико седластих елемената има у матрици.

Опис улаза

У првом реду дату су два природна броја n и m ($1 \leq n, m \leq 2000$).
У наредних n редова налази се по m целих бројева, раздвојених по једним размаком.

Додатна ограничења- У тест примерима вредним 60 поена ће важити $1 \leq n, m \leq 100$.

Опис излаза

На стандардни излаз исписати један број - укупни број седластих елемената у матрици.

Пример 1

Улаз	*Излаз*	*Објашњење*

`3 4`
`6 9 6 5`
`5 4 3 2`
`7 8 6 5`
	`2`	Седласти елементи су два броја 5 у четвртој колони. Сваки од њих је најмањи у свом реду и истовремено највећи у својој колони, па је укупан број седластих елемената 2.

Пример 2

Улаз	*Излаз*	*Објашњење*

`3 3`
`1 2 3`
`4 5 6`
`9 8 5`
	`0`	У овој матрици не постоји елемент који је истовремено најмањи у свом реду и највећи у својој колони, па је број седластих елемената 0.

Решење

Наивно решење

Прво решење које би могло да нам падне на памет је да прођемо кроз све елементе матрице и за сваки проверимо да ли је највећи у својој колони и најмањи у свом реду. Ако јесте, увећавамо бројач за 1.

Сложеност решења

Ово решење је сложености $O(n \cdot m \cdot (n + m))$ и доноси 60 поена.

```
MAXN = 2000
MAXM = 2000


n, m = map(int, input().split())
a = [list(map(int, input().split())) for _ in range(n)]

broj = 0

for i in range(n):
    for j in range(m):

        # provera da li je najmanji u redu
        najmanji = min(a[i])
        if a[i][j] == najmanji:

            # provera da li je najveći u koloni
            najveci = max(a[k][j] for k in range(n))

            if najveci == a[i][j]:
                broj += 1

print(broj)
```

Решење уз претходно одређивање минимума и максимума по колонама

Претходно решење се лако може унапредити ако приметимо да за сваки ред m пута одређујемо његов минимум, а за сваку колону n пута одређујемо њен максимум. Уместо тога, можемо направити два низа - један у ком памтимо минимуме свих редова и један у коме памтимо максимуме свих колона. Када генеришемо те низове, пролазимо кроз све елементе матрице и за сваки проверавамо да ли је једнак најмањем у свом реду и највећем у својој колони и ако јесте, повећавамо бројач за 1. Пошто смо унапред одредили минимуме по редовима и максимуме по колонама, ова провера је сада једноставна и своди се на 2 поређења по 2 броја.

Сложеност решења

Ово решење је сложености $O(n \cdot m)$ и доноси 100 поена.

```python
n, m = map(int, input().split())

a = [list(map(int, input().split())) for _ in range(n)]

min_u_redu = [min(red) for red in a]

max_u_koloni = [max(a[i][j] for i in range(n)) for j in range(m)]

broj = 0

for i in range(n):
    for j in range(m):
        if a[i][j] == min_u_redu[i] and a[i][j] == max_u_koloni[j]:
            broj += 1

print(broj)
```

Задатак: Наводњавање

Аутори: Филип Марић, Душан Попадић и Александар Николић

Дуж једне улице налазе се паркићи у којима се налази дрвеће. На почетку улице налази се пумпа за воду. Од те пумпе вуку се подземне цеви до сваког појединачног дрвета. Познат је максимални укупан број метара цеви које је могуће поставити у склопу предвиђеног буџета. Са пумпом се повезује редом један по један паркић од почетка улице, све док је то могуће тј. све док се не стигне до првог паркића у коме није могуће повезати сва стабла. Колико стабала ће бити повезано?

Опис улаза

У првом реду стандардног улаза се уноси број паркића n $(1 \leq n \leq 10^5)$. Након тога се, у наредних n редова, за сваки парк учитава растојање од почетка улице (у метрима, највише 10^6) и број стабала (највише 10). У последњем реду се уноси највећа могућа укупна дужина цеви b (највише 10^9) у метрима.

Опис излаза

На стандардни излаз исписати тражени највећи могући број стабала.

Додатна ограничења

Тест примери су подељени у две групе: - у тест примерима вредним 50 поена важи: $n \leq 1000$; - у тест примерима вредним 50 поена нема додатних ограничења;

Пример

Улаз	*Излаз*	*Објашњење*

`5`
`12 5`
`8 3`
`15 4`
`17 2`
`3 8`
`150`
	`16`	Дужина цеви потребна за један паркић једнака је производу растојања тог паркића од пумпе и броја стабала у њему, јер до сваког стабла иде посебна цев од пумпе. Паркићи се повезују редом од најближег ка даљем: за растојања 3, 8 и 12 укупно се потроши $3 \cdot 8 + 8 \cdot 3 + 12 \cdot 5 = 108$ метара, а додавањем следећег паркића премашује се буџет 150 (јер бисмо имали $108 + 15 \cdot 4 = 168$), па се повезује укупно $8 + 3 + 5 = 16$ стабала.

Решење

Пошто се паркови редом наводњавају од најближег ка најдаљем природно је да помислимо како их треба сортирати по удаљености од почетка улице. За сваки парк имамо две информације - удаљеност од почетка улице и број стабала. Ове податке можемо чувати у низу парова где је први елемент пара удаљеност, а други број стабала. Након сортирања овог низа по удаљености, пролазимо редом кроз паркове и проверавамо да ли је преостала дужина цеви довољна да би се наводњавала сва стабла у парку (потребна дужина цеви за парк је производ удаљености од почетка улице и броја стабала). Ако имамо довољно цеви за наводњавање парка, умањимо преосталу дужину цеви и увећамо број наводњеног дрвећа. Стајемо или када смо прошли кроз све паркове или када више немамо довољно цеви за наредни парк.

```python
n = int(input())

parkovi = []

for i in range(n):
    r, b = map(int, input().split())

    parkovi.append((r, b))

l = int(input())

parkovi = sorted(parkovi, key=lambda p: p[0])

br = 0
for p in parkovi:
    if l >= p[0] * p[1]:
        l -= p[0] * p[1]
        br += p[1]
    else:
        break

print(br)
```

Задатак: Заборавна плесачица

Аутор: Теодора Обрадовић

Дуња воли да тренира плес. Иако јој плесање добро иде, она је веома заборавна па понекад заборави опрему за тренинг код куће. Када се то догоди она мора да се врати кући и да узме опрему пре тренинга. Међутим, између њених часова у школи и тренинга нема много времена, па она излази са последњег часа (који се сваког дана завршава у исто време) да би стигла на време да узме опрему и оде на тренинг. Разредна је приметила да Дуња често излази раније са часова и занима је колико се то пута догодило.

Помозите разредној и израчунајте колико пута је Дуња отишла раније са наставе.

Опис улаза

Прва линија стандардног улаза садржи два природна броја h и m који представљају редом, сат и минут у коме се завршава последњи час. У другој линији стандардног улаза уноси се број n - број радних дана од почетка школске године. У наредних n линија стандардног улаза уносе се два броја - сат и минут када је Дуња напустила учионицу.

Опис излаза

На стандардни излаз исписати са колико часова је Дуња изашла раније, тј. колико пута је заборавила опрему.

Пример

Улаз	*Излаз*	*Објашњење*

`18 10`
`5`
`18 10`
`17 49`
`18 12`
`16 49`
`17 0`
	`3`	Првог дана наставе, Дуња је отишла када се завршио час, тако да није изашла раније. Трећег дана наставе, Дуња је изашла пар минута након краја часа - требало јој је времена да спакује књиге, али није изашла раније са часа. Другог, четвртог и петог дана Дуња је изашла пре 18:10. Дакле, три дана је заборављала опрему за плес.

Решење

Како су сва времена дата у облику сата и минута у оквиру истог дана, поређење се најлакше врши тако што се за свако време израчуна колико је минута прошло од поноћи до тог тренутка по формули $brojSati * 60 + brojMinuta$. Пролазимо кроз све Дуњине изласке са часа и проверавамо да ли је укупан број минута од поноћи мањи од укупног броја минута од поноћи до завршетка часа. Ако јесте, бројач повећавамо за 1.

```
h, m = map(int, input().split()) # sat i minut kada se završava poslednji čas
brDana=int(input())

zaboravila = 0

for i in range(brDana):
    hi, mi=map(int, input().split())

    if hi * 60 + mi < h * 60 + m:
        zaboravila+=1

print(zaboravila)
```

Задатак: Већи од свих парних

Аутори: Милан Вугделија и Никола Чутурић

Написати програм који за два дата низа исписује колико има бројева у првом низу, који су већи од **свих парних** елемената другог низа. Уколико у другом низу нема парних елемената, рачунати да је сваки број у првом низу испунио тражени услов да је већи од свих парних елемената другог низа.

Опис улаза

Са стандардног улаза се најпре уноси број елемената првог низа n. У следећем реду уноси се n

целих бројева A_i - елементи првог низа.
Затим се уноси број елемената другог низа m, а у наредном реду m целих бројева B_i - елементи другог низа.

Опис излаза

На стандардни излаз исписати један цео број - број елемената првог низа који су строго већи од свих парних елемената другог низа.

Ограничења- $1 \leq n, m \leq 2 \cdot 10^5$ - $1 \leq A_i, B_i \leq 10^9$

Тест примери су подељени у две групе: - у тест примерима вредним 70 поена важи: $n, m \leq 1000$; - у тест примерима вредним 30 поена нема додатних ограничења.

Пример 1

Улаз	*Излаз*	*Објашњење*

`3`
`3 2 4`
`4`
`2 0 1 3`
	`2`	Парни елементи другог низа су 2 и 0; у првом низу бројеви 3 и 4 су већи од оба парна елемента другог низа, па је одговор 2.

Пример 2

Улаз	*Излаз*	*Објашњење*

`5`
`1 1 1 1 1`
`2`
`5 3`
	`5`	Други низ нема парних елемената, па услов важи за све елементе првог низа, због чега је одговор 5.

Решење

Наивно решење

Очигледно решење овог задатка је да се прође кроз све чланове низа 1 и да се затим за сваки члан прође кроз све чланове низа 2 и провери да ли је члан из низа 1 већи од свих парних у низу 2. Ако јесте повећавамо бројач за 1.

Анализа сложености

Пошто је потребно да се прође кроз сваки пар бројева, сложеност решења је $O(n \cdot m)$. Ово решење доноси 70 поена.

```python
n1 = int(input())
a1 = [int(x) for x in input().split()]
n2 = int(input())
a2 = [int(x) for x in input().split()]

br_vecih = 0
for x in a1:
    veci_od_svih_parnih = True
    for y in a2:
        if y % 2 == 0 and x <= y:
            veci_od_svih_parnih = False
    if veci_od_svih_parnih:
        br_vecih += 1
```

```
print(br_vecih)
```

Одређивање највећег парног у другом низу

Претходно решење се лако може поправити тако што прво одредимо највећи парни елемент у другом низу (ако постоји), а затим прођемо кроз све елементе првог низа и бројимо колико их има да су већи од највећег парног у другом низу. Једини случај на који морамо посебно обратити пажњу је случај када у низу 2 не постоји ниједан парни елемент. У том случају на излаз само исписујемо број елемената првог низа.

Анализа сложености

Потребно је по једном проћи кроз сваки низ, тако да је сложеност решења $O(n + m)$. Ово решење доноси 100 поена.

```
n1 = int(input())
a1 = list(map(int, input().split()))

n2 = int(input())
a2 = list(map(int, input().split()))

maxPar2 = -10**10
postojiParan = False

for i in range(n2):
    if a2[i] % 2 == 0:
        maxPar2 = max(maxPar2, a2[i])
        postojiParan = True

if not postojiParan:
    print(n1)
else:
    brVecih = 0
    for i in range(n1):
        if a1[i] > maxPar2:
            brVecih += 1
    print(brVecih)
```

Задатак: Бриџ

Аутор: Никола Чутурић

Мали Тео воли да игра бриџ. Он би хтео да учествује на бриџ турнирима и да осваја награде на њима иако још увек није много добар играч. Јачина сваког бриџ играча може се описати једним природним бројем који представља ранг играча, а Тео зна да може да се такмичи успешно само против играча који имају мањи ранг од њега. Тео може да приступи ранговима свих регистрованих играча у бриџ савезу и занима га колико играча има ранг мањи од неког природног броја. Како бриџ турнира има много, а Тео има много слободног времена, он би желео одговор на ово питање за више различитих природних бројева. Помозите Теу да добије одговоре на његова питања.

Опис улаза

У првом реду стандардног улаза налази се један природан број N - број регистрованих бриџ играча.

У другом реду стандардног улаза налази се N природних бројева R_i - рангови регистрованих бриџ играча.

У трећем реду стандардног улаза налази се један природан број Q - број упита које Тео поставља.

У наредних Q редова улаза налази се по један природан број Q_i - ранг за који Тео жели да зна колико играча има мањи ранг од Q_i.

Опис излаза

У сваком од Q редова стандардног излаза налази се по један број - у i-том реду излаза налази се број регистрованих бриџ играча чији је ранг мањи од Q_i.

Ограничења- $1 \leq N, Q \leq 10^5$ - $1 \leq R_i, Q_i \leq 10^9$

Постоје две групе тест примера у којима важе додатна ограничења: - у тест примерима вредним 50 поена важи: $N, Q \leq 1000$; - у тест примерима вредним 70 поена важи: $R_i, Q_i \leq 10^6$.

Групе нису дисјунктне, тј. неки у неким тест примерима важе оба додатна ограничења.

Пример

Улаз	*Излаз*	*Објашњење*

`5`
`19 24 5 30 17`
`3`
`20`
`24`
`27`
	`3`
	`3`
	`4`
		Рангови играча су: 19, 24, 5, 30 и 17.
		За упит 20, мањи ранк од 20 имају играчи са ранковима 19, 5 и 17, укупно 3 играча.
		За упит 24, мањи ранк од 24 имају играчи са ранковима 19, 5 и 17, поново 3 играча (24 се не рачуна јер није строго мање).
		За упит 27, мањи ранк од 27 имају играчи са ранковима 19, 24, 5 и 17, укупно 4 играча.

Решење

Наивно решење

Очигледно решење овог задатка је да се за сваки учитани упит прође кроз низ рангова и преброји колико их има који су мањи од учитаног броја.

Анализа сложености

Пошто је потребно да се прође кроз цео низ за сваки упит, сложеност решења је $O(q \cdot n)$.

```
n = int(input())
rangovi = list(map(int, input().split()))

q = int(input())

for _ in range(q):
    x = int(input())
```

```
    br = 0
    for r in rangovi:
        if r < x:
            br += 1
    print(br)
```

Решење коришћењем бинарне претраге

Претходно решење се може поправити уколико сортирамо низ рангова, а онда бинарном претрагом тражимо колико има елемената низа који су мањи од учитаног броја. Ово можемо постићи коришћењем уграђене библиотечке функције `bisect_left`.

Анализа сложености

Сортирање низа је сложености $O(n \log(n))$, а бинарне претраге $O(\log(n))$. Пошто је бинарну претрагу потребну извести q пута, укупна сложеност је $O(n \log(n) + q \log(n))$. Ово решење доноси 100 поена.

```
import bisect

n = int(input())
rangovi = list(map(int, input().split()))

rangovi.sort()

q = int(input())

for _ in range(q):
    trazeniRang = int(input())
    brojManji = bisect.bisect_left(rangovi, trazeniRang)
    print(brojManji)
```

Мапирање решења унапред

Уколико је опсег учитаних бројева који се учитавају упити релативно мали (рецимо до 10^6) ми можемо унапред одредити за сваки могући учитани број које ће бити решење. У низ `resenje` на место i уписујемо колико има бројева у низу рангова који су мањи од i. Ово радимо тако што сортирамо низ рангова и онда пролазимо кроз све бројеве од 1 до 10^6. Када се у низу појављује број до ког смо тренутно стигли, бројач повећавамо за 1 и у наредне елеменете низа уписујемо нову вредност бројача (док се поново број који обрађујемо у том тренутку не појави у низу рангова). Када учитавамо упите x_i, једноставно испишемо $resenje[x_i]$.

Анализа сложености

Сортирање низа је сложености $O(n \log(n))$, а попуњавања низа `resenje` је $O(1)$. Иако се попуњавање низа ради у константној сложености, та константа зависи од тога колики је опсег бројева у оквиру упита који сме да се појави. Ако је тај опсег до 10^6, тај део не утиче превише на време извршавања, али ако је тај опсег око 10^9, попуњавање низа ће трајати превише дуго.

Напомена: У складу са текстом задатка ово решење је требало да доноси 70 поена (подзадатак у коме важи $R_i, Q_i \leq 10^6$). Међутим, Комисија је, грешком, приликом генерисања тест примера *користила погрешну верзију генератора, тако да је у свим тест примерима важило ограничење да се уносе бројеви до* 10^6 *и зато је ово решење на такмичењу доносило 100 поена.*

```
n = int(input())

ranks = list(map(int, input().split()))

q = int(input())

ranks = sorted(ranks)

resenje = [0]*(1000001)

p = 0
x = 0
for i in range(n):
    for j in range(p, ranks[i] + 1):
        resenje[j] = x
    x += 1
    p = ranks[i] + 1

for i in range(q):
    qi = int(input())
    if qi > ranks[-1]:
        print(len(ranks))
    else:
        print(resenje[qi])
```