#	Problem Name	Time Limit	Memory Limit
1	nicknames	1 sec.	64 MB
2	ropes	1 sec.	64 MB
3	superfactorial	1 sec.	64 MB
4	vogons	2 sec.	64 MB
5	universe	2 sec.	64 MB

ამოცანა A. “მეტსახელები”

საქართველოში ზოგი მეტსახელი ადამიანის გვარიდან წარმოიქმნება. მაგალითად, კალაძეს „კალას“ ეძახიან, ბერიძეს კი „ბეროს“.

თქვენი ამოცანაა დაწეროთ პროგრამა, რომელიც ქართული გვარისგან ავტომატურად წარმოქმნის შესაბამის მეტსახელს. ამ ეტაპისთვის პროგრამა ორი ტიპის გვარებს უნდა ამუშავებდეს: რომლებიც „აძე“-თი მთავრდებიან და რომლებიც „იძე“-თი მთავრდებიან. „აძე“-თი დამთავრებული გვარებისგან მეტსახელი „ძე“-ს მოცილებით წარმოიქმნება. „იძე“-თი დამთავრებულ გვარებში ბოლო სამი ასოს „ო“-თი ჩანაცვლება ხდება.

თქვენ მოცემული გაქვთ ლათინური ასოებით ჩაწერილი ქართული გვარი *S*. თუ იგი „adze“ ან „idze“-თი მთავრდება, წარმოქმენით შესაბამისი მეტსახელი და დაბეჭდეთ იგი. წინააღმდეგ შემთხვევაში გამოიტანეთ *S* უცვლელად.

შეზღუდვები

S სტრიქონი ლათინური ანბანის არანაკლებ 5 და არაუმეტეს 20 სიმბოლოსგან შედგება. მისი პირველი ასო ზედა რეგისტრშია.

შემომავალი ფაილის ფორმატი

შესატანი მონაცემების nicknames.in ფაილის ერთადერთ ხაზში წერია სტრიქონი *S*. გარანტირებულია, რომ იგი ამოცანის შეზღუდვებს აკმაყოფილებს.

გამომავალი ფაილის ფორმატი

გამოსატანი მონაცემების nicknames.out ფაილში დაბეჭდეთ მოცემული გვარის შესაბამისი მეტსახელი, თუ იგი „adze“ ან „idze“-თი მთავრდება, წინააღმდეგ შემთხვევაში კი ეს გვარი უცვლელად.

შემომავალი ფაილის მაგალითი (nicknames.in)	გამომავალი ფაილის მაგალითი (nicknames.out)
Manaidze	Manao
Kaladze	Kala
Sarajishvili	Sarajishvili

ამოცანა B. “თოკები”

მანაოს აქვს განსხვავებული სიგრძის რამდენიმე თოკი. იგი ხშირად კვანძავს მათ სხვადასხვა ხერხებით და როგორც იქნა აღმოაჩინა გაკვანძვის იდეალური ალგორითმი! ალგორითმის მსვლელობის პროცესში მანაოს აქვს გადაკვანძული თოკებით მიღებული ჯაჭვი. მას თანდათან ეკვანძება ახალი თოკები ხან ერთი ბოლოდან, ხან მეორედან.

ალგორითმის დასაწყისში ჯაჭვი მანაოს თოკებს შორის ყველაზე გრძელისგან შედგება. პირველ ნაბიჯზე მას მარცხნიდან (თოკის ერთ-ერთი ბოლო პირობითად მარცხენად ჩავთვალოთ) ებმევა დარჩენილ თოკებს შორის უგრძესი. მეორე ნაბიჯზე მას მარჯვნიდან ებმევა შემდეგი უგრძესი თოკი. შემდეგ ამ ჯაჭვს ისევ მარცხნიდან ეკვანძება მორიგი უგრძესი თოკი. ასე გრძელდება მანამ, სანამ ყველა თოკი ამ ჯაჭვში არ გაერთიანდება.

ყოველი თოკი აღინიშნება ინგლისური ანბანის რომელიმე ასოს რამდენიმე კოპიით, მაგალითად „aaa” ან “xxxxx”. ყველა თოკს განსხვავებული ასო შეესაბამება. უფრო გრძელი თოკი უფრო დიდი რაოდენობის სიმბოლოთი აღინიშნება. როდესაც ორი თოკის გაკვანძვა ხდება, შესაბამისი სტრიქონები ერთმანეთს ედგმევა და მარცხენა თოკის ბოლო სიმბოლო და მარჯვენა თოკის პირველი სიმბოლო ადგილებს იცვლიან. მაგალითად, „aaa” და “xxxxx” თოკების გადაბმისას მივიღებთ ჯაჭვს “aaxaxxx”.

ახლა მანაოს სჭირდება, რომ აღწერილი ალგორითმით გადაკვანძული თოკები განკვანძოს, ყველაზე გრძელი მოაშოროს და დარჩენილები კვლავ შეაერთოს ამავე ალგორითმით. დაეხმარეთ მას იმის დადგენაში, თუ რა ჯაჭვი უნდა მიიღოს.

შეზღუდვები

მოცემული ჯაჭვის სიგრძე არ აღემატება 500 სიმბოლოს.

მოცემული ჯაჭვი შედგება არანაკლებ 2 და არაუმეტეს 26 თოკისგან.

ჯაჭვის ყოველი თოკი ინგლისური ანბანის ქვედა რეგისტრის არანაკლებ 3 ასოსგან შედგება.

ჯაჭვში შემავალი თოკების სიგრძეები განსხვავებულია.

შემომავალი ფაილის ფორმატი

შესატანი მონაცემების ropes.in ფაილის ერთადერთ სტრიქონში ჩაწერილია მანაოს თოკების მოცემული ალგორითმით გადაკვანძვის შედეგად მიღებული თოკების ჯაჭვი. გარანტირებულია, რომ ეს ჯაჭვი ამოცანის შეზღუდვებს აკმაყოფილებს.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა ropes.out ფაილში დაბეჭდეთ სტრიქონი, რომელიც შეესაბამება მოცემული ჯაჭვის განკვანძვის, უგრძესი თოკის მოშორების და დარჩენილების ხელმეორედ გადაკვანძვით მიღებულ ჯაჭვს.

შემომავალი ფაილის მაგალითი (ropes.in)	გამომავალი ფაილის მაგალითი (ropes.out)
aaacacccbcbb	bbabaaa
aacacccdcddddbdbbb	bbbcbcccacaa

განმარტება.

პირველ მაგალითში ჯაჭვის განკვანძვის შემდეგ მანაოს დარჩება „aaaa", "ccccc" და "bbb" თოკები. მათ შორის ყველაზე გრძელია „ccccc"და მისი მოშორების შემდეგ რჩება "aaaa" და "bbb". მოცემული ალგორითმით გადაკვანძვის შემდეგ მიიღება ჯაჭვი „bbabaaa".

ამოცანა C. "სუპერფაქტორიალი"

შემოვიღოთ მთელი დადებითი X რიცხვის X!! სუპერფაქტორიალის ცნება შემდეგი რეკურენტული ფორმულით:

X!! = 1, თუ X=1.
*X!! = (X-1)!! * X!, თუ X>1.*

X! ჩანაწერი X რიცხვის ჩვეულებრივ ფაქტორიალს აღნიშნავს, ანუ X! = X * (X-1) * (X-2) * … * 1. მაგალითად, 5! = 5*4*3*2*1 = 120. თავის მხვრივ, განსაზღვრებიდან გამომდინარე, 5!! = 4!! * 5! = 3!! * 4! * 5! = … = 1 * 2! * 3! * 4! * 5!

თქვენ მოცემული გაქვთ მთელი რიცხვი *N*. გამოთვალეთ, თვლის ათობით სისტემაში რიცხვი *N!!* რამდენი მიყოლებითი ნულით ბოლოვდება.

შეზღუდვები

1 ≤ N ≤ 1,000,000,000

შემომავალი ფაილის ფორმატი

შესატანი მონაცემების superfactorial.in ფაილის ერთადერთ ხაზში ჩაწერილია ერთი მთელი რიცხვი *N*. გარანტირებულია, რომ ის ამოცანის შეზღუდვებს აკმაყოფილებს.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა superfactorial.out ფაილში ჩაწერეთ ერთადერთი მთელი რიცხვი - *N!!* რიცხვის ბოლოში მიყოლებით მდგომი 0-ების რაოდენობა.

შემომავალი ფაილის მაგალითი (superfactorial.in)	გამომავალი ფაილის მაგალითი (superfactorial.out)
6	2
50	262

განმარტება.
პირველ მაგალითში 6!! = 24883200. ეს რიცხვი 2 ნულით ბოლოვდება.

მეორე მაგალითში 50!! რიცხვი იმდენად დიდია, რომ მის ჩაწერას ამ გვერდის დარჩენილი ადგილი არ ეყოფა.

ამოცანა D. “ვოგანები, ყველგან ვოგანები”

დედამიწა საფრთხეშია! ვოგანების ფლოტი უკვე შეიკრიბა და ისინი სადაცაა მარსს გამოსცდებიან. გალაქტიკის პრეზიდენტის, ზაპჰოდ ბიბელბროქსის ბრძანებით, უნდა აშენდეს ახალი ჰიპერსივრცული მაგისტრალი, რომელიც პირდაპირ ჩვენს პლანეტაზე გაივლის. ნორმალური ადამიანი ახლა იფიქრებდა, რომ ამოცანა დედამიწის გადარჩენაზეა... ჰა ჰა, თქვენ შეცდით! ეს მდაბიო არსებების პლანეტა სულ მალე მოისპობა!

თქვენ ხართ პროგრამისტი ვოგანების ერთ-ერთ ხომალდზე და გევალებათ ფლოტის დასაშვებად ზედაპირის გასწორების დაგეგმვა. ადგილი, სადაც ფლოტი დაშვებას აპირებს, N რაოდენობის მიყოლებითი უბნისგან შედგება. i-ურ უბანზე ზედაპირის სიმაღლე A_i მეტრს შეადგენს. ყოველი უბნის ნიადაგი შეიძლება სხვადასხვა ტიპის იყოს და ამიტომ სხვადასხვა უბნის სიმაღლის შეცვლა სხვადასხვა ოდენობის ენერგიას მოითხოვდეს. კერძოდ, i-ური უბნის ზედაპირის სიმაღლის 1 მეტრით შესაცვლელად B_i ოდენობის ენერგია არის საჭირო. შევნიშნოთ, რომ ვოგანებს აქვთ ტექნიკური შესაძლებლობა როგორც უბნის ზედაპირის დონის შესამცირებლად, ასევე ასამაღლებლად.

ზედაპირის გასწორება შემდეგი სქემითაა დაგეგმილი. უნდა შეირჩეს მიყოლებითი უბნების მიმდევრობა k, k+1, …, k+m ($0 \le m$) და რაიმე მთელი რიცხვი H და შემდეგ ყოველი შერჩეული უბნის სიმაღლე ვოგანების ჰიპერულტრაიარაღის გამოყენებით H-ის ტოლი გახდეს. ამის გასაკეთებლად E ოდენობის ენერგიაა გამოყოფილი. საჭიროა, რომ რაც შეიძლება გრძელი უბნების მიმდევრობა შეირჩეს, რომლის გასწორებას ეს ენერგია ეყოფა. ასევე ვოგანებს აინტერესებთ, რა მაქსიმალური ოდენობის ენერგიის შენარჩუნებას მოახერხებენ ასეთი უგრძესი მიმდევრობის გასწორების შემდეგ.

შეზღუდვები

$1 \le N \le 8,000$

$0 \le A_i, B_i \le 10^7$ ყოველი i-სთვის $[1, N]$ დიაპაზონში. ყველა ეს რიცხვი მთელია.

$0 \le E \le 10^{18}$, E მთელი რიცხვია.

შემომავალი ფაილის ფორმატი

შესატანი მონაცემების vogons.in ფაილის პირველი ხაზი შეიცავს ორ მთელ რიცხვს N და E. მეორე ხაზზე წერია თითო ჰარით გამოყოფილი N რაოდენობის მთელი რიცხვი - A მიმდევრობა. მესამე ხაზზე იგივე ფორმატში B მიმდევრობაა ჩაწერილი. გარანტირებულია, რომ ეს ინფორმაცია ამოცანის შეზღუდვებს აკმაყოფილებს.

გამომავალი ფაილის ფორმატი

გამოსატანი მონაცემების vogons.out ფაილში დაბეჭდეთ ორი მთელი რიცხვი - მიყოლებითი უბნების მაქსიმალური რაოდენობა, რომელთა გასწორება მოხერხდება მოცემული ენერგიით და მაქსიმალური ენერგია, რომელიც მათი გასწორების შემდეგ დარჩება.

შემომავალი ფაილის მაგალითი (vogons.in)	გამომავალი ფაილის მაგალითი (vogons.out)
3 100 1 2 1 1 101 100	2 99
3 100 1 2 3 1 100 99	3 0
5 7 0 2 2 0 1 2 3 0 3 6	4 1

განმარტება.

პირველ მაგალითში ენერგიის 100 ერთეული მხოლოდ 2 მიყოლებითი უბნის გასწორებას ეყოფა: ან პირველი ორი უბანი, ან ბოლო ორი უბანი H=2 სიმაღლეზე უნდა გავასწოროთ. პირველ გეგმაში 1 ერთეული ენერგია იხარჯება, ხოლო მეორეში კი 100.

მეორე მაგალითში სამივე უბანი შეგვიძლია H=2 სიმაღლეზე გავასწოროთ, რითაც ენერგიის მარაგს ამოვწურავთ. შევნიშნოთ, რომ ამ უბნების მაგალითად H=0 სიმაღლეზე დასაყვანად დაიხარჯებოდა 496 ერთეული ენერგია, ხოლო H=3 სიმაღლეზე გასასწორებლად 102 ერთეული ენერგია.

მესამე მაგალითში ოპტიმალურია პირველი ოთხი უბნის H=0 სიმაღლეზე გასწორება.

ამოცანა E. "სიცოცხლე, სამყარო და საერთოდ ყველაფერი"

ცნობილი ფაქტია, რომ პასუხი სიცოცხლეზე, სამყაროზე და საერთოდ ყველაფერზე არის 42. მისი ორობითი ჩანაწერია 101010, ფოტონს სჭირდება 10^{-42} წამი პროტონის დიამეტრის გასავლელად, ხოლო წყალზე 42 გრადუსით დაცემული სხივი კი ცისარტყელას წარმოქმნის. შეიძლება კიდევ უამრავი საინტერესო ფაქტის აღნიშვნა, მაგრამ ჩვენს ამოცანაში ეს უკვე აღარაა აქტუალური.

თუმცა ხანდახან სამყაროში შეცდომები იპარება და ისეთი რაღაცეები გვხვდება, როგორიცაა მაგალითად 6x9=54... არადა, რათქმაუნდა, 42 უნდა იყოს. მიუხედავად ამისა, დაკვირვებული ადამიანი შენიშნავდა, რომ 13-ობით სისტემაში მართლაც 6x9=42.

მოდით ცოტა განვაზოგადოთ ეს მოვლენა. ჩვენ შეგვიძლია ვთქვათ, რომ სამყაროში შეცდომა არ არის, თუ რიცხვის ჩანაწერი რომელიმე K-ობით თვლის სისტემაში ქვესტრიქონად შეიცავს 42-ს (მაგალითად 47 (base 10) = 142 (base 5) და შესაბამისად რიცხვი 47 სამყაროში შეცდომას არ ქმნის).

თქვენ გეძლევათ ორი მთელი ათობით სისტემაში ჩაწერილი A და B რიცხვი. გამოთვალეთ X = $A*B$ და დაადგინეთ, რამდენ თვლის სისტემაში შეიცავს X რიცხვის ჩანაწერი ქვესტრიქონად 42-ს. თუ ასეთი თვლის სისტემა არ მოიძებნა, გამოიტანეთ "bug".

შეზღუდვები

$1 \le A, B \le 1,000,000,000$

A და B მთელი რიცხვებია.

შემომავალი ფაილის ფორმატი

შესატან მონაცემთა universe.in ფაილის ერთადერთ სტრიქონში ჩაწერილია ორი A და B რიცხვი.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა universe.out ფაილში დაბეჭდეთ სიტყვა „bug", თუ X=$A*B$ რიცხვი არც ერთი თვლის სისტემაში არ შეიცავს 42-ს ქვესტრიქონად. წინააღმდეგ შემთხვევაში დაბეჭდეთ, სულ რამდენ თვლის სისტემაში შეიმჩნევა ეს მოვლენა.

შემომავალი ფაილის მაგალითი (universe.in)	გამომავალი ფაილის მაგალითი (universe.out)
6 9	1
2 31	2
1 1	bug

განმარტება.

პირველი მაგალითი. 6*9=54 მხოლოდ 13-ობით სისტემაში შეიცავს ქვესტრიქონად 42-ს.

მეორე მაგალითი. 2*31=62. 15-ობით სისტემაში 62 ჩაიწერება როგორც 42, ხოლო 6-ობითში კი როგორც 142.

მესამე მაგალითი. 1*1=1. ეს რიცხვი ნებისმიერ თვლის სისტემაში 1-ის ტოლია.