#	Problem Name	Time Limit	Memory Limit
1	push	1 sec.	64 MB
2	matrix	1 sec.	64 MB
3	greentao	1 sec.	64 MB
4	robot	1 sec.	64 MB
5	manaobros	2 sec.	64 MB

ამოცანა A. “მიწოლა”

მანაო თამაშობს ძალიან მარტივ თამაშს. გარკვეულ პოზიციაში დევს ყუთი და მან ეს ყუთი უნდა მიაგოროს დანიშნულების ადგილამდე. სათამაშო სიბრტყეზე შემოვიღოთ მართკუთხა კოორდინატები. ყუთის საწყისი პოზიცია არის წერტილი (X_1, Y_1), ხოლო ბოლოს იგი (X_2, Y_2) წერტილში უნდა მოხვდეს. თვითონ სიბრტყე უსასრულოა.

ყუთის გადაადგილების ერთადერთი საშუალება მიწოლაა. ერთ მიწოლაში მანაო ყუთს ზუსტად K ერთეულით გადაადგილებს ჰორიზონტალურად ან ვერტიკალურად. თუ რომელიმე მიწოლის პროცესში ან ბოლოში ყუთი მოხვდება დანიშნულების წერტილში, თამაში დამთავრდება.

დაადგინეთ, შეუძლია თუ არა მანაოს თამაშის დამთავრება და თუ შეუძლია, მაშინ მინიმუმ რამდენი მიწოლა დასჭირდება საამისოდ.

შეზღუდვები

$-100 \le X_1, Y_1, X_2, Y_2 \le 100$

$1 \le K \le 10$

ყუთის საწყისი და დანიშნულების ადგილი განსხვავებულია.

შემომავალი ფაილის ფორმატი

შესატანი მონაცემების push.in ფაილის ერთადერთი ხაზი შეიცავს ხუთ მთელ რიცხვს X_1, Y_1, X_2, Y_2, K.

გამომავალი ფაილის ფორმატი

გამოსატანი მონაცემების push.out ფაილში დაბეჭდეთ 0, თუ მანაოს არ შეუძლია ყუთის დანიშნულების ადგილამდე მიტანა. წინააღდეგ შემთხვევაში დაბეჭდეთ ამისთვის საჭირო მიწოლების მინიმალური შესაძლო რაოდენობა.

შემომავალი ფაილის მაგალითი (push.in)	გამომავალი ფაილის მაგალითი (push.out)
2 0 4 3 3	2
-1 -1 1 1 3	0

განმარტება.

პირველ მაგალითში მანაოს შეუძლია ყუთს ჯერ „ზემოთ” მიაწვეს, რითაც იგი (2, 3) წერტილში მოხვდება, ხოლო შემდეგ მარჯვნივ დაიწყოს მისი გადაადგილება, რის პროცესშიც ყუთი (4, 3) წერტილში მოხვდება და თამაში მორჩება.

მეორე მაგალითში მანაო ვერ მოახერხებს, ყუთი დანიშნულების ადგილის ვერტიკალზე ან ჰორიზონტალზე მიიყვანოს.

ამოცანა B. "მატრიცა"

ჩვეულებრივი ადამიანისთვის სიტყვა „მატრიცა" ცნობილ ფილმთან ასოციირდება. პროგრამირებაში ასე მრავალგანზომილებიან მასივებს ეძახიან. მათემატიკაში „მატრიცა" ორგანზომილებიან ცხრილს წარმოადგენს, რომლის თითოეული ჩანაწერი (ასევე ცნობილი როგორც ელემენტი) შეიძლება რიცხვი, სიმბოლო ან მათემატიკური გამოსახულებაც კი იყოს.

მატრიცას, რომელშიც სტრიქონების და სვეტების რაოდენობა ერთმანეთის ტოლია, კვადრატული ეწოდება. კვადრატულ მატრიცებში გამოყოფენ რამდენიმე სახეობას. მათ შორის ზოგიერთი გახლავთ:

a) ნულოვანი (Zero). ნულოვანი მატრიცის ყველა ელემენტი 0-ის ტოლია.

b) ერთეულოვანი (Identity). ეს ისეთი მატრიცაა, რომლის მთავარ დიაგონალზე ყველა ელემენტი 1-ის ტოლია, დანარჩენები კი 0-ის. მთავარი დიაგონალი შეიცავს ყველა ელემენტს, რომლის სტრიქონის და სვეტის ნომერი ემთხვევა.

c) სიმეტრიული (Symmetric). ასეთ მატრიცაში ყოველი ელემენტი ტოლია თავისი სიმეტრიული ელემენტის მთავარი დიაგონალის მიმართ. ელემენტი თუ არის i-ური სტრიქონის და j-ური სვეტის გადაკვეთაზე, მაშინ მისი სიმეტრიული ელემენტი j-ური სტრიქონის და i-ური სვეტის გადაკვეთაზე არის.

d) ანტისიმეტრიული (Antisymmetric). ამ მატრიცაში ყოველი ელემენტი მოდულით ტოლია მთავარი დიაგონალის მიმართ თავისი სიმეტრიული ელემენტის, ხოლო ნიშნით საპირისპიროა.

თქვენ მოცემული გაქვთ მატრიცა N სტრიქონით და N სვეტით. დაადგინეთ, იგი რომელიმე სახეობას თუ მიეკუთვნება. შეამჩნევდით, რომ მატრიცა თუ ნულოვანია, მაშინ იგი აუცილებლად სიმეტრიულიც და ანტისიმეტრიულიცაა და ასევე თუ მატრიცა ერთეულოვანია, მაშინ იგი აუცილებლად სიმეტრიულია. ამიტომ ნულოვანი და ერთეულოვანი მატრიცებისთვის (ანტი)სიმეტრიულობის შესახებ დაწერა საჭირო აღარაა.

შეზღუდვები

$1 \le N \le 100$

მატრიცის ყოველი ელემენტი მთელი რიცხვია [-1000, 1000] შუალედში.

შემომავალი ფაილის ფორმატი

შესატანი მონაცემების matrix.in ფაილის პირველ სტრიქონში ჩაწერილია ერთი მთელი რიცხვი N. შემდეგი N სტრიქონიდან თითოეული შეიცავს N ცალ რიცხვს - მატრიცის ელემენტებს.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა matrix.out ფაილში დაბეჭდეთ ერთი სიტყვა, რომელიც მატრიცის სახეობას განსაზღვრავს: Zero, Identity, Symmetric, Antisymmetric ან None, თუ მატრიცა არც ერთი სახეობის არაა ზემოთ ჩამოთვლილთაგან.

შემომავალი ფაილის მაგალითი (matrix.in)	გამომავალი ფაილის მაგალითი (matrix.out)
2 0 0 0 0	Zero
2 1 0 0 1	Identity
3 1 1 1 1 1 1 1 1 1	Symmetric
3 0 2 -3 -2 0 1 3 -1 0	Antisymmetric
2 1 2 3 4	None

ამოცანა C. “გრინ-ტაოს თეორემა”

გრინ-ტაოს თეორემის თანახმად, მარტივ რიცხვთა მიმდევრობაში ნებისმიერი სიგრძის არითმეტიკული პროგრესიები არსებობს. სხვა სიტყვებით რომ ვთქვათ, ნებისმიერი ნატურალური K რიცხვისთვის არსებობს არითმეტიკული პროგრესია K ცალი წევრით, ყოველი რომელთაგანი მარტივ რიცხვს წარმოადგენს.

ცხადია, ამ დებულების სისწორე ეფუძნება იმას, რომ მარტივი რიცხვების მიმდევრობა უსასრულოა. ჩვენ კი ვცადოთ სასრულ ინტერვალებზე ვიპოვოთ მარტივი რიცხვებისგან შემდგარი არითმეტიკული პროგრესიები.

თქვენ გეძლევათ ორი ნატურალური რიცხვი L და R. დაადგინეთ ყველაზე გრძელი არითმეტიკული პროგრესიის წევრების რაოდენობა, რომლის თითოეული წევრი მარტივი რიცხვია $[L, R]$ შუალედიდან.

შეზღუდვები

$1 \le L \le R \le 1{,}000{,}000{,}000$

$R - L \le 5000$

შემომავალი ფაილის ფორმატი

შესატანი მონაცემების greentao.in ფაილის ერთადერთ ხაზში წერია ორი ნატურალური რიცხვი L და R.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა greentao.out ფაილში ჩაწერეთ ერთი მთელი რიცხვი: საჭირო არითმეტიკული პროგრესიის წევრების მაქსიმალური რაოდენობა.

შემომავალი ფაილის მაგალითი (greentao.in)	გამომავალი ფაილის მაგალითი (greentao.out)
1 15	3

განმარტება.
[1, 15] შუალედზე მარტივი რიცხვები გახლავთ 2, 3, 5, 7, 11 და 13. ჩვენი პასუხია არითმეტიკული პროგრესია პირველი წევრით 3 და სხვაობით 2.

ამოცანა D. "რობოტი"

გვაქვს N x M განზომილებების მქონე დაფა. დაფის ზემო მარცხენა უჯრაში დგას რობოტი, რომელიც შეგვიძლია ვმართოთ და ოთხი მიმართულებით (ზემოთ, ქვემოთ, მარცხნივ, მარჯვნივ) გადავაადგილოთ. კონკრეტული მიმართულებით გადაადგილების ბრძანების მიღებისას რობოტი ამოწმებს, ამ მიმართულებით ერთი უჯრედის გავლით იგი დაფის ფარგლებს ხომ არ დატოვებს. თუ რობოტი დაფაზე დარჩება, მაშინ იგი შესაბამისი მიმართულებით ერთი უჯრედით გადაადგილდება, თუ არა - რჩება თავის ადგილზე.

ბრძანებები მოკლედ ჩაიწერება მათი ინგლისური დასახელებების პირველი ასოების მეშვეობით - 'L' აღნიშნავს მარცხნივ გადაადგილებას, 'R' - მარჯვნივ, 'U' - ზემოთ და 'D' - ქვემოთ. ამრიგად, ბრძანებების მიმდევრობა ჩაიწერება ამ ოთხი ასოსგან შემდგარი სტრიქონით. მაგალითად, „LURR" სტრიქონი ნიშნავს, რომ რობოტს ჯერ მარცხნივ, შემდეგ ზემოთ და შემდეგ 2-ჯერ მარჯვნივ გადაადგილებას ვუბრძანებთ.

ჩვენ შეგვიძლია რაღაც ბრძანებების მიმდევრობა ჩავციკლოთ, ანუ გადავცეთ რობოტს უსასრულოდ. მაგალითად, თუ „LURR" ბრძანებათა მიმდევრობას ჩავციკლავთ, რობოტი შეასრულებს ბრძანებების უსასრულო მიმდევრობას „LURRLURRLURRLURR…". ჩვენი მიზანია, რობოტი დაფის ყოველ უჯრედში ერთხელ მაინც მოვახვედროთ. დაადგინეთ ისეთი უმოკლესი ბრძანებათა მიმდევრობა, რომლის ჩაციკვლისას ამ მიზანს მივაღწევთ.

შეზღუდვები

$1 \leq N, M \leq 50;\ N * M > 1$

შემომავალი ფაილის ფორმატი

შესატან მონაცემთა robot.in ფაილის ერთადერთი ხაზი შეიცავს ორ მთელ რიცხვს N და M.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა robot.out ფაილის ერთადერთ ხაზში დაბეჭდეთ უმოკლესი ბრძანებათა მიმდევრობა, რომლის ჩაციკვლისას რობოტი საბოლოო ჯამში შემოივლის მთელ დაფას. თუ ასეთი მიმდევრობა რამდენიმეა, გამოიტანეთ ის, რომელსაც ლექსიკოგრაფიულად მინიმალური სტრიქონი შეესაბამება. ორი სტრიქონის ლექსიკოგრაფიულად შედარებისას უნდა მოძებნოთ პირველი პოზიცია, რომელზეც ამ სტრიქონებს განსხვავებული სიმბოლოები აქვთ და შეადაროთ ეს სიმბოლოები ანბანის მიხედვით.

შემომავალი ფაილის მაგალითი (robot.in)	გამომავალი ფაილის მაგალითი (robot.out)
2 3	DRU

განმარტება.

რობოტი ასრულებს უსასრულო ბრძანებათა მიმდევრობას „DRUDRUDRU…”. გადავნომროთ დაფის უჯრედები ზემოდან ქვემოთ 1-დან N-მდე და მარცხნიდან მარჯვნივ 1-დან M-მდე. რობოტი დაიწყებს (1, 1) უჯრედში და ყოველი ბრძანების შემდეგ ამ უჯრედებში აღმოჩნდება:

1. D – (2, 1)
2. R – (2, 2)
3. U – (1, 2)
4. D – (2, 2)
5. R – (2, 3)
6. U – (1, 3)
7. D – (2, 3)
8. R – (2, 3)
9. U – (1, 3)
10. D – (2, 3)
11. R – (2, 3)
12. U – (1, 3)

და ასე შემდეგ. როგორც ხედავთ, მე-6 ბრძანების შესრულების მერე რობოტს მთელი დაფა აქვს შემოვლილი.

ამოცანა E. “Manao Bros”

გამოვიდა ახალი თამაში „Manao Bros”. მასში სულ N ტურია, ყოველ რომელთაგანზე მანაოს უწევს ბევრი ხტუნვა, სროლა და ქულების დაგროვება. i-ურ ტურზე მაქსიმუმ X_i ქულის მოგროვება შეიძლება.

ტურების გავლა მიყოლებით არ ხდება. პირველი ტურის გავლის შემდეგ მოთამაშეს ეძლევა არჩევანი, თუ რომელ ტურზე გააგრძელოს თამაში. ამ ტურის გავლის შემდეგ იგი ისევ არჩევნის წინაშე დგება. ზოგ შემთხვევაში მას შეიძლება ისეთი ტურიც შესთავაზონ, რომელიც მან უკვე გაიარა. ამ შემთხვევაში მას შეუძლია ამ ტურის მეორედ გავლა, მაგრამ ქულას ამისთვის მეორედ ვეღარ აიღებს.

ტურები, რომლებსაც მოთამაშეს შესთავაზებენ თამაშის გასაგრძელებლად, შემთხვევითად არ გენერირდება. დავნომროთ თამაშის ტურები მთელი რიცხვებით 1-დან N-მდე. თამაში ყოველთვის ტურზე ნომრით 1 იწყება. როდესაც მოთამაშე გადის ტურს ნომრით i, მას სთავაზობენ ასარჩევად ტურების S_i სიმრავლეს. ეს სიმრავლე არ იცვლება მაშინაც, თუ შესაბამისი ტური მოთამაშემ მეორედ ან მესამედ დახურა.

თამაში მთავრდება მაშინ, როდესაც მოთამაშე გადის ტურს ნომრით N. ზოგჯერ შეიძლება, იგი მოხვდეს ისეთ ტურზე, საიდანაც ტური ნომრით N აღარ მიიღწევა. გამოთვალეთ, რა მაქსიმალური შესაძლო ჯამური ქულის აღება შეიძლება თამაშის დამთავრებისას.

შეზღუდვები

$2 \le N \le 100,000$

$0 \le X_i \le 1000$ ყოველი i-სთვის 1-დან N-მდე

პირველი ტურიდან ნებისმიერი ტური მიიღწევა.

S_i სიმრავლეებში შემავალი ელემენტების საერთო რაოდენობა არ აღემატება 200,000-ს.

შემომავალი ფაილის ფორმატი

შესატან მონაცემთა manaobros.in ფაილის პირველ სტრიქონში ჩაწერილია ერთი მთელი N რიცხვი. შემდეგ სტრიქონში წერია N ცალი ერთმანეთისგან ჰარებით გამოყოფილი რიცხვი - $X_1, X_2, ..., X_N$. შემდეგი (N-1) სტრიქონიდან i-ური შეიცავს ინფორმაციას i-ური ტურის გავლის შემდეგ შემოთავაზებული ტურების S_i სიმრავლის შესახებ. იგი იწყება დადებითი რიცხვით K_i - ამ სიმრავლეში შემავალი ელემენტების რაოდენობით. შემდეგ წერია K_i ცალი განსხვავებული ტურის ნომერი, რომლებზე გადასვლა შეიძლება i-ური ტურის გავლის შემდეგ.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა manaobros.out ფაილში ჩაწერეთ მოთამაშის მაქსიმალური შესაძლო ქულა თამაშის დამთავრებისას.

შემომავალი ფაილის მაგალითი (manaobros.in)	გამომავალი ფაილის მაგალითი (manaobros.out)
4 3 10 2 4 2 2 3 1 2 2 1 4	9
5 0 1 2 3 4 4 5 2 4 3 1 1 1 1 1 1	10

განმარტება.

პირველ მაგალითში თამაშში სულ 4 ტურია, რომლებზეც შესაბამისად 3, 10, 2 და 4 ქულის მოგროვებაა შესაძლებელი. პირველი ტურის მერე მოთამაშეს შეუძლია აირჩიოს, მეორე ტურზე გააგრძელოს თუ მესამეზე. მეორე ტურის შემდეგ მას ისევ მეორე ტურს სთავაზობენ, ამიტომ ასე იგი თამაშს ვერ დაამთავრებს. მესამე ტურის შემდეგ შეიძლება პირველ ტურზე დაბრუნება ან მეოთხე ტურზე გადასვლა.

მეორე მაგალითში პირველი ტურიდან შეიძლება ნებისმიერ სხვა ტურზე გადასვლა, ხოლო მათი გავლის შემდეგ ისევ პირველ ტურთან დაბრუნება. თუ მოთამაშე მე-N ტურს ბოლოსთვის მოიტოვებს, იგი ყველა ტურის ქულას აიღებს.