თეორიული ინფორმატიკა

დავალება #2

ამ დავალებაში თქვენი მიზანი იქნება დაწეროთ პროგრამა რომელიც ორ ლენტიანი თიურინგის მანქანისთვის იპოვეის მის შესაბამის ერთ ლენტიან მანქანას და გააკეთებს მის სიმულაციას. ორ ლენტიანი თიურინგის მანქანები, რომლებიც ამ დავალებაში გექნებათ მოცემული შეზღუდული იქნება მხოლოდ 3 სიმბოლოთი - "0", "1" და "_", რომელიც ცარიელ უჯრას აღნიშნავს. მეორეს მხრივ, თქვენს მიერ აგებულ მანქანას, მეტი სიმბოლოს გამოყენება შეუძლია მათ შორის ნებისმიერი ციფრის, ნებისმიერი დიდი ლათინური ასოს (პატარები - არ შეიძლება) და ზოგიერთი სიმბოლოს (მაგალითად იგივე "_", რომელიც ისევ ცარიელ ადგილებს აღნიშნავს ლენტაზე). რეალურად შეგიძლიათ ASCII ცხრილის 48-ე სიმბოლოდან ("0") დაწყებული 95-ეს ჩათვლით ("_") ნებისმიერი სიმბოლოები გამოიყენოთ.

პროგრამირების ენა ამ დავალებაშიც თქვენ თვითონ შეგიძლიათ აირჩიოთ, შედარებით გავრცელებული (და ლინუქსზე რომ ადვიალად ყენდება/ეშვება ისეთი) პროგრამირების ენებიდან. ხშირად იყენებენ ხოლმე Python3 (ყველაზე ხშირად), Python2, Java ან C++-ს მაგრამ თუ გინდათ სხვა ენაც შეგიძლიათ გამოიყენოთ. უბრალოდ წინასწარ გადაამოწმეთ ჩემთან რომ მაგ ენის კომპილატორი/ინტერპრეტატორი მქონდეს. თქვენი პროგრამა მონაცემებს უნდა კითხულობდეს standard input-დან და უნდა წერდეს standard output-ში.

ერთ ლენტიან მანქანად გადაკეთება [3 ქულა]

ამ ნაწილში თქვენს პროგრამას standard input-დან (ქვემოთ მოცემული ფორმატით) გადაეცემა ორ ლენტიანი (დეტერმინისტული) თიურინგის მანქანის აღწერა.

აღწერა დაიწყება ხაზით რომელზეც წერია ერთი რიცხვი n - მანქანის მდგომარეობების რაოდენობა. მდგომარეობების დანომრვა წინა დავალებასავით 0-დან იწყება და მდგომარეობა #0 ყოველთვის იქნება საწყისი მდგომარეობა, ხოლო მდგომარეობა $\#(n-1)$ იქნება მიმღები მდგომარეობა. უარმყოფ მდგომარეობას ამ მანქანაში არ გავითვალისწინებთ და ჩავთვლით რომ თუ რომელიმე მდგომარეობიდან, რომელიმე სიმბოლოთი გადასვლა არ გვაქვს, ეს იგივეა რაც უარმყოფ მდგომარეობაში გადასვლა.

შემდეგ გვექნება $n-1$ ცალი ხაზი, რომლებიც თანმიმდევრულად აღწერენ თითო მდგომარეობიდან შესაძლო გადასვლებს. i-ური ხაზი დაიწყება ერთი რიცხვი m_i-თი (რადგან მანქანა დეტერმინისტულია: $0 \leq m_i \leq 9$), რომელიც ამ მდგომარეობიდან შესაძლო გადასვლების რაოდენობას აღნიშნავს. ამის შემდეგ ამავე ხაზზე იქნება $7m_i$ ცალი ცარიელი ადგილით გამოყოფილი ნაწყვეტი, თითო გადასვლისთვის 7 ნაწყვეტი:

1. პირველი ლენტიდან რა სიმბოლო უნდა წავიკითხოთ ამ გადასვლისთვის
2. მეორე ლენტიდან რა სიმბოლო უნდა წავიკითხოთ
3. რომელ მდგომარეობაში (ნომრით) გადავდივართ ამ გადასვლით
4. პირველ ლენტაზე რა სიმბოლოს ვწერთ
5. მეორე ლენტაზე რა სიმბოლოს ვწერთ
6. პირველ ლენტაზე საით მივდივართ ("L" ან "R")
7. და მეორე ლენტაზე საით მივდივართ ("L" ან "R")

მაგალითად ავტომატი რომელიც მიიღებს ნებისმიერ "0"-ზე დაყყებულ სიტყვას შეგვიძლია შემდეგნაირად ავღწეროთ:

$M^{(2)}_{0\Sigma^*}$

```
2
1 0 _ 1 0 _ R R
```

თქვენი მიზანია ააგოთ ერთ ლენტიანი თიურინგის მანქანა, რომელიც იგივე ენას მიიღებს და გამოიტანოთ მსგავსი ფორმატით standard output-ში. ეს ფორმატი ისევ ერთი რიცხვით დაიყვება, რომელიც მდგომარეობების რაოდენობა იქნება და შემდეგ თითო ხაზზე თითო მდგომარეობის აღწერა იქნება, მაგრამ თითო გადასვლისთვის $7m_i$ ნაწყვეტის მაგივრად, $4m_i$ ნაწყვეტი დაგვჭირდება:

1. ლენტიდან რა სიმბოლო უნდა წავიკითხოთ ამ გადასვლისთვის
2. რომელ მდგომარეობაში (ნომრით) გადავდივართ ამ გადასვლით
3. ლენტაზე რა სიმბოლოს ვწერთ
4. ლენტაზე საით მივდივართ (“L” ან “R”)

მაგალითად წინა ავტომატის ექვივალენტური ერთ ლენტიანი ავტომატი შეიძლება შემდეგნაირად აღიწეროს:

$M^{(1)}_{0\Sigma^*}$

```
2
1 0 1 0 R
```

აქ პირველი მანქანა მეორე ლენტას არაფერში არ იყენებდა და ამიტომ მარტივად გადაკეთდა იგივე რაოდენობის მდგომარეობიან ერთ ლენტიან ავტომატად, თუმცა ზოგადად უფრო მეტი წვალება შეიძლება დაგჭირდეთ. ორივე ლენტის სიმულაციისთვის სასარგებლოა ერთ ლენტაზე ორივე ლენტის შიგთავსის დატანა და წამკითხავი მიმთითებლის პოზიციების აღნიშნვა შესაბამისი ახალი სიმბოლოებით.

თქვენი მანქანა აუცილებლად საწყისი ორ-ლენტიანი მანქანის ექვივალენტური უნდა იყოს. ამისი შემოწმება არაგადაწყვეტადი ამოცანაა, ამიტომ რამდენიმე შემთხვევითად შერჩეულ input-ზე გავუშვებ და ორივე მანქანამ (საწყისმა და გადაკეთებულმა) იგივე პასუხები უნდა გამოიტანოს. აგებული მანქანის ზომა საწყისთან შედარებით რამდენჯერმე დიდი იქნება, მაგრამ წრფივად უნდა იზრდებოდეს და 50-მდე მდგომარეობის მქონე მანქანებისთვის 20000-ზე მეტ მდგომარეობიან მანქანებს არ გვაძლევდეს. ასევე, თქვენი მანქანის ოპერაციების რაოდენობა (მუშაობის დრო) არ უნდა იყოს საჭყისის მუშაობის დროის კვადრატზე ასიმპტოტურად უარესი. ეს ყველაფერი ზუსტად არ შემოწმდება, მაგრამ თუ თქვენი მანქანა ძალიან ნელა იმუშავებს (მაგალითად ექსპონენციალურ დროში), ის მოცემულ დროში ტესტებს ვერ გაივლის.

ამ ნაწილში დაწერილ პროგრამას დაარქვით “convert” და პროგრამირების ენაზე დამოკიდებული გაფართოება (მაგალითად “convert.py”)

ერთ ლენტიანი მანქანის სიმულაცია [3 ქულა]

ამ ნაწილში თქვენ უნდა დაწეროთ წინა ნაწილის ფორმატით აღწერილი ერთ ლენტიანი თიურინგის მანქანის სიმულაცია. standard input-დან შემოგივათ ერთ ლენტიანი თიურინგის მანქანა და დამატებით ერთ ხაზზე ჩაწერილი input სიტყვა, რომელიც შედგენილი იქნება მხოლოდ “0”-ების და “1”-ებისგან (თუმცა მანქანის ოპერაციები შეიძლება ლენტაზე სხვა სიმბოლოებსაც წერდნენ)

თქვენი მიზანია გააკეთოთ ამ პროგრამის სიმულაცია და ყოველი ნაბიჯის შემდეგ ცალკე ხაზზე გამოიტანოთ თუ რომელ (რა ნომერ) მდგომარეობაში იქნება ეს ავტომატი ამ ნაბიჯის მერე. თუ მანქანა უარმყოფ მდგომარეობაში გადავიდა, მდგომარეობის ნომრად “-1” უნდა გამოიტანოთ. ლენტაზე არსებული კონფიიგურაციის გამოტანა არ არის საჭირო, მაგრამ თქვენთვის უნდა დაიმახსოვროთ, რომ შემდეგ ნაბიჯებზე რომელი გადასვლები გაკეთოთ იმის გადაწყვეტა შეგეძლოთ.

მაგალითად მოცემულია ავტომატი და მისი სტრინგზე მუშაობის შედეგი:

input

```
2
2 0 0 0 R 1 1 1 R
0011
```

output

```
0
0
1
```

აქ პასუხი სულ 3 ხაზს შეიცავს, მიუხედავად იმისა რომ 4 სიმბლოიან სიტყვას ვაძლევთ, იმიტომ რომ 3 ოპერაციის შემდეგ ავტომატი მუშაობას ამთავრებს და დანარჩენ სიტყვას არც უყურებს.

ამ ნაწილში დაწერილ პროგრამას დაარქვით “simulate” და პროგრამირების ენაზე დამოკიდებული გაფართოება (მაგალითად “simulate.py”)

დავალების გამოგზავნა

დაწერილი ორი პროგრამა, სწორი სახელებით მოათავსეთ ერთ folder-ში, რომელსაც სახელად უნდა დაარქვათ თქვენი ელ-ფოსტის პრეფიქსი (@ სიმბოლომდე რაც არის) და უნდა შეკუმშოთ zip ფაილად. მიღებული ფაილი ატვირთეთ ქლასრუმზე შესაბამისი დავალების სექციაში.