თეორიული ინფორმატიკა

დავალება #1

ამ დავალებაში თქვენი მიზანი იქნება დაწეროთ პროგრამა რომელიც შეამოწმებს ერგება თუ არა მოცემულ ტექსტს მოცემული რეგულარული გამოსახულება. ამოცანა ორ ნაწილად იქნება დაყოფილი:

1. პირველ ნაწილში მოცემული იქნება რეგულარული გამოსახულება, რომელიც თქვენ უნდა გაადააკეთოთ სპეციფიური ტიპის ექვივალენტურ არადეტერმინისტულ სასრულ ავტომატად
2. მეორე ნაწილში უნდა დაწეროთ პროგრამა, რომელიც ამ არადეტერმინისტული ავტომატის სიმულაციას გააკეთებს და შედეგს დააბრუნებს.

არადეტერმინისტული ავტომატის სიმულაციის დროს, მისი დეტერმინისტულ ავტომატად გადაკეთება არ არის საჭირო. უბრალოდ ყოველ ნაბიჯზე დაიმახსოვრეთ მდგომარეობების რა სიმრავლეში არის შესაძლებელი მოცემულ ეტაპზე არადეტერმინისტული ავტომატი რომ იყოს და თავისთავად გამოგივათ დეტერმინისტული ავტომატის სიმულაციაც.

პროგრამირების ენა შეგიძლიათ თქვენ თვითონ 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 შეგიძლიათ ეს დავალება რამე ახალი ენის სასწავლად გამოიყენოთ და იმ ენაზე დაწეროთ პროგრამა. თუ თქვენი ენა ამ სიაში არ არის, ჩემთან გადაამოწმეთ დასაშვებია თუ არა მისი გამოყენება (როგორც წესი კომპილირება/გაშვების ინსტრუქცია მჭირდება ხოლმე რომ დავუშვა). თქვენი პროგრამა მონაცემებს უნდა კითხულობდეს standard input-დან და უნდა წერდეს standard output-ში.

გამოსახულების ავტომატად გადაკეთება [3 ქულა]

ამ ნაწილში თქვენს პროგრამას standard input-დან გადაეცემა ერთი ხაზი, რეგულარული გამოსახულების აღწერით. სიმარტივისთვის ამ რეგულარულ გამოსახულებაში დასაშვები სიმბოლოები მხოლოდ ლათინური ანბანის პატარა ასოები და ციფრები იქნება. სხვა სიმბოლოებით შედგენილი გამოსახულებების გათვალისწინება არ გჭირდებათ. გამოსახულებაში ასევე იქნება სიმბოლოები “|” - რომელიც გაერთიანებას აღნიშნავს და “*” - რომელიც მის წინ მდგომი გამოსხულების ნებისმიერ რაოდენობაჯერ გამეორებას აღნიშნავს. კონკატენაცია უბრალოდ გამოსახულებების თანმიმდევრობით ჩაწერით აღინიშნება. ცხადია ასევე უნდა გაითვალისწინოთ ფრჩხილები - “(” და “)”. მაგალითად, ტიპიური გამოსახულება რომელიც ყველა ოპერაციას შეიცავს შეიძლება იყოს: `(ab*c(0|1)*)*`, რაც მათემატიკური ჩანაწერით $(ab^*c(0 \cup 1)^*)^*$-ს შეესაბამება.

გამოსახულებებში არ გვაქვს ცარიელი სტრინგის აღმნიშვნელი სპეციალური სიმბოლო - ϵ , მაგრამ ამისთვის შეგვიძლია უბრალოდ ცარიელი ფრჩხილები `()` გამოვიყენოთ.

თქვენ უნდა ააგოთ არადეტერმინისტული სასრული ავტომატი, რომლიც ამ გამოსახულებას შეესაბამება და თან რამდენიმე შეზღუდვას დაექვემდებარება. პირველ რიგში, ამ ავტომატს უნდა ჰქონდეს არაუმეტეს გამოსახულების სიგრძეზე ერთით მეტი მგდომარეობა. დამატებით, ეს ავტომატი არ უნდა შეიცავდეს ე.წ. ϵ-გადასვლებს. ϵ გადასვლების მოშორება მდგომარეობების რაოდენობის გაზრდის გარეშე საკმაოდ მარტივია და პრობლემა არ უნდა შეგიქმნათ. ამის გაკეთების ერთი გზა არის ყველა მდგომარეობა, რომელიც მიმღებში გადადიოდა ϵ-ით, ასევე მიმღები გავხადოთ. დამატებით კიდევ, ყოველთვის როდესაც A მდგომარეობა ϵ-ით B-ში გადადიოდა, ამ გადასვლის მაგივრად, A-დან ყველა ის გადასვლაც გავაკეთოთ რაც B-დან კეთდებოდა.

თქვენმა პროგრამამ მიღებული ავტომატის აღწერა ქვემოთ მოცემული ფორმატით უნდა გამოიტანოს standard output-ში:

პირველ ხაზზე უნდა დაწეროთ 3 მთელი რიცხვი n, a და t (space-ებით გამოყოფილი). n არის მდგომარეობების რაოდენობა, a - მიმღები მდგომარეობების რაოდენობა და t -

გადასვლების რაოდენობა (აქ თითო სიმბოლოთი გადასვლა ცალკე გადასვლად ითვლება). იგულისხმება რომ მდგომარეობები დანომრილია 0-დან დაწყებული მთელი რიცხვებით და საწყისი მდგომარეობა ყოველთვის არის მდგომარეობა ნომრით 0. მეორე ხაზზე a ცალი მთელი რიცხვი უნდა გამოიტანოთ, რომლებიც მიმღები მდგომარეობების ინდექსებს უნდა აღნიშნავდნენ. ამის შედეგ თითოეული მდგომარეობისთვის უნდა მოდიოდეს ერთ ხაზზე ჩაწერილი შემდეგი ინფორმაცია: ხაზის პირველი რიცხვი (k_i) უნდა მიუთითებდეს თუ რამდენი გადასვლა გვაქვს ამ მდგომარეობიდან (ამ k_i-ების ჯამი ყოველთვის t-ს ტოლი უნდა იყოს, შემდეგ კი (space-ებით გამოყოფილი) k_i ცალი წყვილი სადაც წყვილის პირველი ელემენტი იქნება სიმბოლო რომლითაც გადასვლა კეთდება და მეორე ელემენტი იქნება რიცხვი, რომელიც იმ მდგომარეობის ნომერს აღნიშნავს სადაც გადავდივართ.

მაგალითისთვის მოცემულია რამდენიმე რეგულარული გამოსახულება და მათი პასუხი. ცხადია, არ არის აუცილებელი რომ თქვენმა პროგრამამ ზუსტად ეს პასუხი გამოიტანოს, მაგრამ გამოტანილი ავტომატის ენა ამ ავტომატის ენას უნდა ემთხვეოდეს. ამის შესამოწმებლად ყველაზე მარტივი გზა არის რამდენიმე სხვადასხვა სიტყვაზე გადაამოწმოთ ორივეს მუშაობის შედეგი. რეალური შემოწმებაც ამ მეთოდით მოხდება.

input 1

```
(a|b)*(c|())
```

output 1

```
2 2 3
0 1
3 a 0 b 0 c 1
0
```

input 2

```
(ab*c(0|1)*)*
```

output 2

```
3 2 6
0 2
1 a 1
2 b 1 c 2
3 0 2 1 2 a 1
```

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

ავტომატის სიმულაცია [3 ქულა]

ამ ნაწილში თქვენი დაწერილი პროგრამა პირველ ხაზზე უნდა კითხულობდეს შესამოწმებელ ტექსტს (სტრინგს) და მეორე ხაზიდან დაწყებული უნდა კითხულობდეს წინა ნაწილის პასუხის ფორმატის მიხედვით გამოტანილ ავტომატს. პროგრამის მიზანია ამ ავტომატის ამ სტრინგზე მუშაობის სიმულაცია გააკეთოს. პროგრამამ პასუხად უნდა გამოიტანოს შემავალი სტრინგის სიგრძის ტოლი სტრინგი, რომლის i-ურ პოზიციაზე მითითებული იქნება შემავალი სტრინგის პირველი i სიმბოლოს წაკითხვის შემდეგ პროგრამა ერთ მაინც მიმღებ მდგომარეობაში იმყოფებოდა თუ არა. იმ პოზიციებზე, როდესაც ავტომატი ერთ მაინც მიმღებ მდგომარეობაში იყო უნდა გამოიტანოთ სიმბოლო “Y”, ხოლო პოზიციებზე რომლებისთვისაც ავტომატი არცერთ მიმღებ მდგომარეობაში არ იყო - სიმბოლო “N”.

ამ სიმულაციისთვის პროგრამას არ უნდა სჭირდებოდეს $\mathcal{O}((N+n)\cdot n)$ ოპერაციაზე მეტი. სადაც N შემავალი სტრინგის სიგრძეს აღნიშნავს, n კიდევ ავტომატის მდგომარეობების რაოდენობას. ამის შესამოწმებლად კოდში არ ჩავიხედები, მაგრამ პროგრამის მუშაობის დროზე დაწესებული იქნება შეზღუდვა მონაცემების ზომის მიხედვით. პროგრამამ ამ დროს

არ უნდა გადააჭარბოს. დროს საკმაოდ არამკაცრად ვითვლი და ამიტომ 2x - 3x შენელებები კოდში სავარაუდოდ არ იქნება პრობლემა, თუმცა 20x და მეტი ალბათ უკვე იქნება.

ყოველი შემთხვევისთვის აქაც მოცემულია შემავალი ინფორმაციის და პასუხის რამდენიმე მაგალითი:

input 1

```
aababacab
2 2 3
0 1
3 a 0 b 0 c 1
0
```

output 1

```
YYYYYYYNN
```

input 2

```
abbc1acabbbbc001cabc
3 2 6
0 2
1 a 1
2 b 1 c 2
3 0 2 1 2 a 1
```

output 2

```
NNNYYNYNNNNNYYYYNNNN
```

ამ ნაწილში დაწერილ პროგრამას დაარქვით “run” და პროგრამირების ენაზე დამოკიდებული გაფართოება (მაგალითად “run.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.