შეტყობინება

აიშა და ბასმა არიან მეგობრები, რომლებიც ერთმანეთთან მიმოწერას აწარმოებენ. აიშას აქვს შეტყობინება M, რომელიც არის S რაოდენობის ბიტის (ანუ, ნულების და ერთების) მიმდევრობა, რომელიც მან უნდა გაუგზავნოს ბასმას. აიშას კომუნიკაცია ბასმასთან ხდება **პაკეტების** გაგზავნით. პაკეტი არის 31 ბიტის მიმდევრობა, დანომრილი ინდექსებით 0-დან 30-მდე. აიშას სურს გაუგზავნოს ბასმას M შეტყობინება მისთვის რაღაც რაოდენობის პაკეტების გაგზავნით.

სამწუხაროდ, აიშას და ბასმას კომუნიკაცია ხელში კლეოპატრას ჩაუვარდა და მას შეუძლია პაკეტების **დაზიანება** . კერძოდ, თითოეულ პაკეტში კლეოპატრას შეუძლია ზუსტად 15 ინდექსზე ბიტების შეცვლა. ანუ, არსებობს 31 სიგრძის C მასივი, რომელშიც თითოეული ელემენტი არის 0 ან 1, შემდეგი შინაარსით:

- $C[i] = 1$ მიუთითებს, რომ კლეოპატრას შეუძლია შეცვალოს ბიტი ინდექსით i. ასეთ ინდექსზე ვიტყვით, რომ კლეოპატრა **აკონტროლებს** მას;
- $C[i] = 0$ მიუთითებს, რომ კლეოპატრას არ შეუძლია შეცვალოს ბიტი ინდექსით i.

მასივი C შეიცავს ზუსტად 15 ცალ ერთიანს და 16 ცალ ნულიანს. M შეტყობინების გაგზავნისას კლეოპატრას მიერ გაკონტროლებული ინდექსების სიმრავლე იგივე რჩება ყველა პაკეტისთვის. აიშამ ზუსტად იცის, რომელ 15 ინდექსს აკონტროლებს კლეოპატრა. ბასმამ მხოლოდ ის იცის, რომ კლეოპატრა აკონტროლებს 15 ინდექსს, მაგრამ მან არ იცის რომელ ინდექსებს.

ვთქვათ აიშამ გადაწყვიტა გაგზავნოს პაკეტი A (რომელსაც ვუწოდოთ **ორიგინალი პაკეტი**). ვთქვათ, B არის ბასმასთან მისული პაკეტი (რომელსაც ვუწოდოთ **დაზიანებული პაკეტი**). ყოველი i -სთვის, სადაც $0 \leq i < 31$:

- თუ კლეოპატრა არ აკონტროლებს ბიტს ინდექსით i ($C[i] = 0$), მაშინ ბასმა მიიღებს ბიტს i შეუცვლელად ($B[i] = A[i]$);
- წინააღმდეგ შემთხვევაში, თუ კლეოპატრა აკონტროლებს ბიტს ინდექსით i ($C[i] = 1$), მაშინ $B[i]$-ს მნიშვნელობას ირჩევს კლეოპატრა.

ყოველი პაკეტის გაგზავნის შემდეგ აიშა მაშინვე გებულობს როგორია შესაბამისი დაზიანებული პაკეტი.

მას შემდეგ, რაც აიშა გააგზავნის ყველა პაკეტს, ბასმა იღებს ყველა დაზიანებულ პაკეტს **იგივე თანმიმდევრობით, როგორც იყვნენ გაგზავნილი** და მან უნდა აღადგინოს საწყისი M შეტყობინება.

თქვენი დავალებაა შეიმუშაოთ და განახორციელოთ სტრატეგია, რომლის საშუალებითაც აიშას შეეძლება ბასმასთვის M შეტყობინების გაგზავნა ისე, რომ ბასმამ შეძლოს დაზიანებული

პაკეტებიდან M შეტყობინების აღდგენა. კერძოდ, თქვენ უნდა მოახდინოთ ორი პროცედურის იმპლემენტაცია. პირველი პროცედურა ასრულებს აიშას მოქმედებებს. მას გადაეცემა M შეტყობინება და C მასივი და მან უნდა გააგზავნოს პაკეტები რათა გადასცეს ინფორმაცია ბასმას. მეორე პროცედურა ასრულებს ბასმას მოქმედებებს. მას ეძლევა დაზიანებული პაკეტები და მან უნდა აღადგინოს საწყისი M შეტყობინება.

იმპლემენტაციის დეტალები

პირველი პროცედურა, რომლის იმპლემენტაციაც უნდა მოახდინოთ, შემდეგია:

```
void send_message(std::vector<bool> M, std::vector<bool> C)
```

- M: S სიგრძის მასივი, რომელიც აღწერს აიშას მიერ ბასმასთვის გასაგზავნ შეტყობინებას;
- C: 31 სიგრძის მასივი, რომელიც აღწერს კლეოპატრას მიერ გაკონტროლებული ბიტების ინდექსებს;
- პროცედურა შეიძლება გამოძახებულ იქნას **არაუმეტეს 2100-ჯერ** ყველა ტესტისთვის.

ამ პროცედურამ პაკეტის გასაგზავნად უნდა გამოიძახოს შემდეგი პროცედურა:

```
std::vector<bool> send_packet(std::vector<bool> A)
```

- A: ორიგინალი პაკეტი (31 სიგრძის მასივი), რომელიც აღწერს აიშას მიერ გაგზავნილ ბიტებს;
- ეს პროცედურა აბრუნებს დაზიანებულ B პაკეტს, რომელიც აღწერს ბასმას მიერ მიღებულ ბიტებს;
- ეს პროცედურა შეიძლება გამოძახებულ იქნას არაუმეტეს 100-ჯერ `send_message`-ის ყოველ გამოძახებაში.

მეორე პროცედურა, რომლის იმპლემენტაციაც უნდა მოახდინოთ, შემდეგია:

```
std::vector<bool> receive_message(std::vector<std::vector<bool>> R)
```

- R: მასივი, რომელიც აღწერს დაზიანებულ პაკეტებს. პაკეტები წარმოშობილია აიშას მიერ ერთ `send_message`-ის გამოძახებიდან და არიან მოცემული **იგივე თანმიმდევრობით, როგორც იყვნენ გაგზავნილი** აიშას მიერ. R-ის ყოველი ელემენტი არის 31 სიგრძის მასივი, რომელიც აღწერს დაზიანებულ პაკეტს;
- ამ პროცედურამ უნდა დააბრუნოს S ბიტიანი მასივი, რომელიც საწყისი M შეტყობინების ტოლია;
- ეს პროცედურა შეიძლება გამოძახებულ იქნას **რამდენჯერმე** ყოველი ტესტისთვის, **ზუსტად ერთხელ** ყოველი შესაბამისი `send_message`-ის გამოძახებისთვის. **თანმიმდევრობა** `receive_message` **პროცედურების გამოძახებისა** არაა აუცილებლად იგივე, როგორც შესაბამისი `send_message`-ების გამოძახებისა.

შევნიშნოთ, რომ გრადერის სისტემაში `send_message` და `receive_message` პროცედურების გამოძახება ხდება **ცალკე პროგრამებში**.

შეზღუდვები

- $1 \leq S \leq 1024$
- C შეიცავს ზუსტად 31 ელემენტს, 16 მათგანი არის 0 და 15 მათგანი არის 1.

ქვეამოცანები და ქულები

თუ რომელიმე ტესტში პროცედურა `send_packet`-ის გამოძახება არღვევს ზემოთ აღწერილ წესებს ან `receive_message` პროცედურის მიერ დაბრუნებული მნიშვნელობა არასწორია, თქვენი ამოხსნის ქულა იმ ტესტში იქნება 0.

წინააღმდეგ შემთხვევაში, ვთქვათ Q არის `send_packet` პროცედურის გამოძახებების მაქსიმალური რაოდენობა `send_message`-ის გამოძახებებს შორის ყველა ტესტისთვის. ასევე, ვთქვათ X არის:

- 1, თუ $Q \leq 66$
- 0.95^{Q-66}, თუ $66 < Q \leq 100$

მაშინ, ქულა ითვლება შემდეგნაირად:

ქვეამოცანა	ქულა	დამატებითი შეზღუდვები
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	დამატებითი შეზღუდვების გარეშე.

აღვნიშნოთ, რომ ზოგჯერ გრადერის მოქმედება შეიძლება იყოს **ადაპტიური**. ეს ნიშნავს, რომ `send_packet`-ის მიერ დაბრუნებული მნიშვნელობები შეიძლება დამოკიდებული იყოს არა მხოლოდ მის შემავალ არგუმენტებზე, არამედ ბევრ სხვა რამეზე, როგორიცაა ამ პროცედურის წინა გამოძახებების შესატანი და დაბრუნებული მნიშვნელობები და გრადერის მიერ გენერირებული ფსევდო-შემთხვევითი რიცხვები. გრადერი არის **დეტერმინისტული** იმ გაგებით, რომ თუ თქვენ მას ორჯერ გაუშვებთ და ორივეჯერ გააგზავნით ერთიდაიმავე პაკეტებს, ის მათში მოახდენს იგივე ცვლილებებს.

მაგალითი

განვიხილოთ შემდეგი გამოძახება:

```
send_message([0, 1, 1, 0],
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

აიშა ცდილობს ბასმას გაუგზავნოს შეტყობინება $[0, 1, 1, 0]$. კლეოპატრას არ შეუძლია შეცვალოს ბიტები ინდექსებით 0-დან 15-მდე, თუმცა მას შეუძლია შეცვალოს ბიტები ინდექსებით 16-დან 30 -მდე.

მაგალითისთვის დავუშვათ, რომ კლეოპატრა ავსებს მის გაკონტროლებულ თანმიმდევრულ ბიტებს მონაცვლეობით 0-ებით და 1-ებით, ანუ ის აწერს 0-ს მის მიერ გაკონტროლებულ პირველ ბიტს (ჩვენს შემთხვევაში ბიტი ინდექსით 16), 1-ს მის მიერ გაკონტროლებულ მეორე ბიტს (ინდექსი 17), 0 -ს მის მიერ გაკონტროლებულ მესამე ბიტს (ინდექსი 18) და ასე შემდეგ.

აიშამ შეიძლება გადაწყვიტოს გაგზავნოს საწყისი შეტყობინების ორი ბიტი ერთ პაკეტში შემდეგნაირად: ის გზავნის პირველ ბიტს პირველ 8 ინდექსში და მეორე ბიტს შემდეგ 8 ინდექსში.

აიშა ამის შემდეგ აგზავნის შემდეგ პაკეტს:

```
send_packet([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

შევნიშნოთ, რომ კლეოპატრას შეუძლია ბოლო 15 ინდექსის ბიტების შეცვლა, ამიტომ აიშას შეუძლია მათი ნებმისმიერად არჩევა, რადგან ისინი მაინც გადაიწერება. კლეოპატრას აღწერილი სტრატეგიით, პროცედურა დააბრუნებს შემდეგს: $[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]$.

აიშა წყვეტს M-ის ბოლო ორი ბიტის გაგზავნას მეორე პაკეტში წინას მსგავსად:

```
send_packet([1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

კლეოპატრას აღწერილი სტრატეგიით პროცედურა დააბრუნებს შემდეგს: $[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]$.

აიშას შეუძლია კიდევ პაკეტების გაგზავნა, თუმცა მან არჩია არ გააგზავნოს.

ამის შემდეგ გრადერი გამოიძახებს შემდეგ პროცედურას:

```
receive_message([[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                  0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
                 [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                  0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]])
```

ბასმა აღადგენს M შეტყობინებას შემდეგნაირად: ყოველი პაკეტიდან ის იღებს პირველივე ბიტს, რომელიც ზედიზედ ორჯერ გვხვდება და ბოლო ბიტს, რომელიც ზედიზედ ორჯერ გვხვდება. ანუ, პირველი პაკეტიდან ის იღებს ბიტებს $[0, 1]$ და მეორე პაკეტიდან ბიტებს $[1, 0]$. მათი შეერთებით ის აღადგენს შეტყობინებას $[0, 1, 1, 0]$, რაც სწორი პასუხია `receive_message`-ის ამ გამოძახებისთვის.

შესაძლებელია ვაჩვენოთ, რომ კლეოპატრას აღწერილი სტრატეგიისთვის და 4 სიგრძის შეტყობინებისთვის ბასმას ეს მეთოდი სწორად აღადგენს M-ს, რაც არ უნდა იყოს C. თუმცა, ეს არაა სწორი ზოგად შემთხვევაში.

სანიმუშო გრადერი

სანიმუშო გრადერი არაა ადაპტიური. კლეოპატრა მის მიერ გაკონტროლებულ ბიტებს ავსებს მონაცვლეობით 0 და 1 ბიტებით, როგორც ზემოთაა აღწერილი.

შეტანის ფორმატი: **პირველი სტრიქონი შეიცავს T რიცხვს, რომელიც აღნიშნავს სცენარების რაოდენობას.** მას მოსდევს T სცენარი. თითოეული მათგანი მოცემულია შემდეგი ფორმატით:

```
S
M[0]  M[1]  ...  M[S-1]
C[0]  C[1]  ...  C[30]
```

გამოტანის ფორმატი: სანიმუშო გრადერი ბეჭდავს შედეგებს T სცენარისთვის იგივე თანმიმდევრობით, როგორც მოცემულია შეტანაში შემდეგი ფორმატით:

```
K L
D[0]  D[1]  ...  D[L-1]
```

აქ, K არის `send_packet`-ის გამოძახებათა რაოდენობა, D არის `receive_message`-ის მიერ დაბრუნებული შეტყობინება და L მისი სიგრძეა.