Μήνυμα

Η Aisha και η Basma είναι δύο φίλες που ανταλλάζουν μηνύματα μεταξύ τους. Η Aisha έχει ένα μήνυμα M, το οποίο είναι μια ακολουθία από S bits (δηλαδή, μηδενικά ή άσσοι) και επιθυμεί να το στείλει στη Basma. Η Aisha επικοινωνεί με τη Basma στέλνοντάς της πακέτα. Ένα πακέτο είναι μια ακολουθία από 31 bits με θέσεις (indexed) από το 0 έως το 30. Η Aisha θέλει να στείλει το μήνυμα M στη Basma στέλνοντάς της κάποιο αριθμό από **πακέτα**.

Ένα πακέτο είναι μια ακολουθία από 31 bits με θέσεις (indexed) από το 0 έως το 30. Η Aisha θέλει να στείλει το μήνυμα M στη Basma στέλνοντάς της κάποιο αριθμό από πακέτα.

Δυστυχώς, η Κλεοπάτρα προσπαθεί να εμποδίσει την επικοινωνία μεταξύ της Aisha και της Basma καθώς μπορεί να **μολύνει** τα πακέτα. Σε κάθε πακέτο η Κλεοπάτρα μπορεί να αλλάξει bits σε ακριβώς 15 θέσεις. Συγκεκριμένα, υπάρχει ένας πίνακας C μήκους 31, στον οποίο κάθε στοιχείο είναι είτε 0 είτε 1, με την ακόλουθη σημασία:

- $C[i] = 1$ υποδηλώνει ότι το bit με θέση i μπορεί να αλλαχτεί από την Κλεοπάτρα. Αυτές τις θέσεις τις ονομάζομαι **ελεγχόμενες** από την Κλεοπάτρα.
- $C[i] = 0$ υποδηλώνει ότι το bit στη θέση i δεν μπορεί να αλλαχθεί από την Κλεοπάτρα.

Ο πίνακας C περιέχει ακριβώς 15 άσσους και 16 μηδενικά. Κατά την αποστολή του μηνύματος M, το σύνολο των θέσεων που ελέγχει η Κλεοπάτρα παραμένει το ίδιο για όλα τα πακέτα. Η Aisha γνωρίζει ακριβώς ποιες 15 θέσεις ελέγχονται από την Κλεοπάτρα. Η Basma ξέρει μόνο ότι 15 θέσεις ελέγχονται από την Κλεοπάτρα, αλλά δεν γνωρίζει ποιες είναι αυτές οι θέσεις.

Έστω A ένα πακέτο που αποφασίζει να στείλει η Aisha (το οποίο ονομάζουμε **αρχικό πακέτο**). Έστω B το πακέτο που λαμβάνεται από την Basma (το οποίο ονομάζουμε **μολυσμένο πακέτο**).

Για κάθε i,τέτοιο ώστε $0 \leq i < 31$:

- εάν η Κλεοπάτρα δεν ελέγχει το bit στη θέση i ($C[i] = 0$), η Basma λαμβάνει το bit i όπως το έστειλε η Aisha ($B[i] = A[i]$),
- διαφορετικά, εάν η Κλεοπάτρα ελέγχει το bit στη θέση i ($C[i] = 1$), η τιμή του $B[i]$ αποφασίζεται από την Κλεοπάτρα.

Αμέσως μετά την αποστολή κάθε πακέτου, η Aisha μαθαίνει ποιο είναι το αντίστοιχο μολυσμένο πακέτο.

Αφού η Aisha στείλει όλα τα πακέτα, η Basma λαμβάνει όλα τα μολυσμένα πακέτα **με τη σειρά που στάλθηκαν** και πρέπει να ανακατασκευάσει το αρχικό μήνυμα M.

Το καθήκον σας είναι να επινοήσετε και να εφαρμόσετε μια στρατηγική που επιτρέπει στην Aisha να στείλει το μήνυμα M στη Basma, έτσι ώστε η Basma να μπορεί να ανακτήσει το μήνυμα M από τα μολυσμένα πακέτα.

Συγκεκριμένα, θα πρέπει να εφαρμόσετε δύο διαδικασίες. Η πρώτη διαδικασία εκτελεί τις ενέργειες της Aisha. Δέχεται ένα μήνυμα M και τον πίνακα C, και πρέπει να στείλει κάποια πακέτα για να μεταφέρει το μήνυμα στη Basma. Η δεύτερη διαδικασία εκτελεί τις ενέργειες της Basma. Δέχεται τα μολυσμένα πακέτα και πρέπει να ανακτηθεί το αρχικό μήνυμα M.

Λεπτομέρειες Υλοποίησης

Η πρώτη διαδικασία που πρέπει να υλοποιήσετε είναι:

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

- M: ένας πίνακας μήκους S που περιγράφει το μήνυμα που θα στείλει η Aisha στη Basma.
- C: ένας πίνακας μήκους 31 που δείχνει τις θέσεις των bits που ελέγχονται από την Κλεοπάτρα.
- Αυτή η διαδικασία μπορεί να κληθεί **το πολύ 2100 φορές** σε κάθε test case.

Αυτή η διαδικασία θα καλεί την ακόλουθη διαδικασία για την αποστολή ενός πακέτου:

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

- A: ένα αρχικό πακέτο (πίνακας μεγέθους 31) που αντιπροσωπεύει τα bits που στέλνει η Aisha.
- Αυτή η διαδικασία επιστρέφει ένα μολυσμένο πακέτο B που αντιπροσωπεύει τα bits που θα ληφθούν από την Basma.
- Αυτή η διαδικασία μπορεί να κληθεί το πολύ 100 φορές σε κάθε κλήση της `send_message`.

Η δεύτερη διαδικασία που πρέπει να υλοποιήσετε είναι:

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

- R: πίνακας που περιγράφει τα μολυσμένα πακέτα. Τα πακέτα προέρχονται από πακέτα που αποστέλλονται από την Aisha με μία κλήση `send_message` και δίνονται **με τη σειρά που**

στάλθηκαν από την Aisha. Κάθε στοιχείο του R είναι ένας πίνακας μήκους 31, που αντιπροσωπεύει ένα μολυσμένο πακέτο.
- Αυτή η διαδικασία θα πρέπει να επιστρέψει έναν πίνακα S αποτελούμενο από S bits που είναι ίσος με το αρχικό μήνυμα M.
- Αυτή η διαδικασία μπορεί να κληθεί πολλές φορές σε κάθε test case, **ακριβώς μία** για την κάθε αντίστοιχη κλήση `send_message`. Η **σειρά των κλήσεων** της διαδικασίας `receive_message` δεν είναι απαραίτητα ίδια με τη σειρά των αντίστοιχων κλήσεων `send_message`.

Σημειώστε ότι στο σύστημα βαθμολόγησης καλούνται οι διαδικασίες `send_message` και `receive_message` σε **δύο ξεχωριστά προγράμματα**.

Περιορισμοί

- $1 \leq S \leq 1024$
- C έχει ακριβώς 31 στοιχεία, από τα οποία τα 16 είναι ίσα με 0 και τα 15 είναι ίσα με 1.

Υποπροβλήματα και βαθμολόγηση

Εάν σε οποιαδήποτε από τα test cases, οι κλήσεις της διαδικασία `send_packet` δεν συμμορφώνονται με τους κανόνες που αναφέρονται πιο πάνω, ή η τιμή επιστροφής οποιασδήποτε από τις κλήσεις στη διαδικασία `receive_message` είναι λάθος, η βαθμολογία της λύσης σας για αυτό το test case θα είναι 0.

Διαφορετικά, έστω Q ο μέγιστος αριθμός κλήσεων της διαδικασίας `send_packet` μεταξύ όλων των κλήσεων του `send_message` σε όλα τα test cases.

Έστω επίσης το X ίσο με:

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

Στη συνέχεια, η βαθμολογία υπολογίζεται ως εξής:

Subtask	Score	Additional Constraints
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	No additional constraints.

Σημειώστε ότι σε ορισμένες περιπτώσεις η συμπεριφορά του δειγματικού βαθμολογητή μπορεί να είναι **προσαρμοστική**. Αυτό σημαίνει ότι οι τιμές που επιστρέφονται από τη `send_packet` μπορεί να μην εξαρτώνται μόνο από τις παραμέτρους εισόδου αλλά και από πολλά άλλα πράγματα, συμπεριλαμβανομένου τις εισόδους και τις επιστρεφόμενες τιμές των προηγούμενων

κλήσεων της διαδικασίας και ψευδο-τυχαίους αριθμούς που δημιουργούνται από τον δειγματικό βαθμολογητή. Ο δειγματικός βαθμολογητής είναι ντετερμινιστικός με την έννοια ότι αν τον τρέξεις δύο φορές με τα ίδια πακέτα θα κάνει τις ίδιες αλλαγές σε αυτά.

Παράδειγμα

Εξετάστε την ακόλουθη κλήση.

```
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])
```

Το μήνυμα που προσπαθεί να στείλει η Aisha στη Basma είναι $[0, 1, 1, 0]$.]. Τα bit με θέσεις από 0 έως 15 δεν μπορεί να τα αλλάξει η Κλεοπάτρα, ενώ τα bit με θέσεις από 16 έως 30 μπορεί να τα αλλάξει η Κλεοπάτρα.

Χάρη παραδείγματος, ας υποθέσουμε ότι η Κλεοπάτρας γεμίζει διαδοχικά bits που ελέγχει με εναλλασσόμενα 0 και 1, δηλ. τοποθετεί 0 στο πρώτο bit που ελέγχει (θέση 16 στην περίπτωσή μας), 1 στο δεύτερο bit που ελέγχει (θέση 17), 0 στο τρίτο bit που ελέγχει (θέση 18) και ούτω καθεξής.

Η Aisha μπορεί να αποφασίσει να στείλει δύο bits από το αρχικό μήνυμα σε ένα πακέτο ως εξής: θα στείλει το πρώτο bit στις 8 πρώτες θέσεις που ελέγχει και το δεύτερο bit στις επόμενες 8 θέσεις που ελέγχει.

Στη συνέχεια, η Aisha επιλέγει να στείλει το ακόλουθο πακέτο:

```
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])
```

Σημειώστε ότι η Κλεοπάτρα μπορεί να αλλάξει bits στις τελευταίες 15 θέσεις, οπότε η Aisha μπορεί να τα ορίσει αυθαίρετα, όπως μπορούν να αντικατασταθούν. Με την υποτιθέμενη στρατηγική της Κλεοπάτρας, η διαδικασία επιστρέφει: $[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]$.

Η Aisha αποφασίζει να στείλει τα δύο τελευταία bits του M στο δεύτερο πακέτο με παρόμοιο τρόπο όπως πριν:

```
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])
```

Με την υποτιθέμενη στρατηγική της Κλεοπάτρας, η διαδικασία επιστρέφει:

$[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Η Aisha μπορεί να στείλει περισσότερα πακέτα, αλλά επιλέγει να μην το κάνει.

Στη συνέχεια, ο βαθμολογητής καλεί την ακόλουθη διαδικασία:

```
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]])
```

Η Basma ανακτά το μήνυμα M ως εξής. Από κάθε πακέτο παίρνει το πρώτο bit που εμφανίζεται δύο φορές στη σειρά και το τελευταίο bit που εμφανίζεται δύο φορές στη σειρά. Δηλαδή από το πρώτο πακέτο παίρνει τα bits $[0,1]$, και από το δεύτερο πακέτο παίρνει τα bits $[1,0]$. Συνδυάζοντάς τα μαζί, ανακτά το μήνυμα $[0,1,1,0]$, που είναι η σωστή επιστροφή για αυτήν την κλήση στο `receive_message`.

Φαίνεται πως με την υποτιθέμενη στρατηγική της Κλεοπάτρας και για μηνύματα μήκους 4, αυτή η προσέγγιση της Basma ανακτά σωστά το M, ανεξάρτητα από την τιμή του C. Ωστόσο, δεν είναι σωστή στη γενική περίπτωση.

Δειγματικός βαθμολογιτής

Ο δειγματικός βαθμολογητής δεν είναι προσαρμοστικός. Αντίθετα, η Κλεοπάτρα γεμίζει διαδοχικά bits που ελέγχει με εναλλασσόμενα 0 και 1 bit, όπως περιγράφεται στο παραπάνω παράδειγμα.

Μορφή εισόδου: **Η πρώτη γραμμή της εισόδου περιέχει έναν ακέραιο αριθμό T, που καθορίζει τον αριθμό των σεναρίων**. Ακολουθούν T σενάρια. Κάθε ένα από αυτά δίνεται στην ακόλουθη μορφή:

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

Μορφή εξόδου: Ο δειγματικός βαθμολογητής γράφει το αποτέλεσμα καθενός από τα σενάρια T με την ίδια σειρά που δόθηκαν στην είσοδο, στην ακόλουθη μορφή:

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

Εδώ, K είναι ο αριθμός των κλήσεων της `send_packet`, D είναι το μήνυμα που επιστρέφεται από τη `receive_message` και L είναι το μήκος του.