Μήνυμα

Η Αϊσά και η Μπάσμα είναι δύο φίλες που αλληλογραφούν μεταξύ τους. Η Αϊσά έχει ένα μήνυμα M, το οποίο είναι μια ακολουθία από S bits (δηλαδή μηδενικά ή άσσοι), το οποίο θέλει να στείλει στην Μπάσμα. Η Αϊσά επικοινωνεί με την Μπάσμα στέλνοντάς της **πακέτα**. Ένα πακέτο είναι μια ακολουθία από 31 bits, με θέσεις από 0 έως 30. Η Αϊσά θέλει να στείλει το μήνυμα M στην Μπάσμα στέλνοντάς της κάποιο αριθμό από πακέτα.

Δυστυχώς, η Κλεοπάτρα προσπαθεί να εμποδίσει την επικοινωνία μεταξύ της Αϊσά και της Μπάσμα και μπορεί να **μολύνει** τα πακέτα. Δηλαδή, 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 στη θέση i μπορεί να αλλαχθεί από την Κλεοπάτρα. Αυτές τις θέσεις τις ονομάζουμε **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 Αϊσά γνωρίζει ακριβώς ποιες 15 θέσεις ελέγχονται από την Κλεοπάτρα. Η Μπάσμα ξέρει μόνο ότι 15 θέσεις ελέγχονται από την Κλεοπάτρα, αλλά δεν ξέρει ποιες είναι αυτές.

Έστω ότι A είναι ένα πακέτο που η Αϊσά αποφασίζει να στείλει (το οποίο ονομάζουμε το **αρχικό πακέτο**). Έστω ότι B είναι το πακέτο που λαμβάνει η Μπάσμα (το οποίο ονομάζουμε το **μολυσμένο πακέτο**). Για κάθε i, τέτοιο ώστε $0 \leq i < 31$:

- αν η Κλεοπάτρα δεν ελέγχει το bit στη θέση i ($C[i] = 0$), η Μπάσμα λαμβάνει το bit i όπως το έστειλε η Αϊσά ($B[i] = A[i]$),
- διαφορετικά, αν 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 Αϊσά μαθαίνει ποιο είναι το αντίστοιχο μολυσμένο πακέτο.

Αφού η Αϊσά στείλει όλα τα πακέτα, η Μπάσμα λαμβάνει όλα τα μολυσμένα πακέτα **με τη σειρά που στάλθηκαν** και πρέπει να ανασυνθέσει το αρχικό μήνυμα M.

Το καθήκον σας είναι να σχεδιάσετε και να υλοποιήσετε μια στρατηγική που θα επιτρέπει στην Αϊσά να στείλει το μήνυμα M στην Μπάσμα, ώστε η Μπάσμα να μπορέσει να ανακτήσει το M

από τα μολυσμένα πακέτα. Συγκεκριμένα, πρέπει να υλοποιήσετε δύο διαδικασίες. Η πρώτη διαδικασία εκτελεί τις ενέργειες της Αϊσά. Δέχεται ένα μήνυμα M και τον πίνακα C και πρέπει να στείλει κάποια πακέτα για να μεταφέρει το μήνυμα στην Μπάσμα. Η δεύτερη διαδικασία εκτελεί τις ενέργειες της Μπάσμα. Δέχεται τα μολυσμένα πακέτα και πρέπει να ανακτεί 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 η Αϊσά θέλει να στείλει στην Μπάσμα.
- C: ένας πίνακας μήκους 31 που υποδεικνύει τις θέσεις των bits που ελέγχονται από την Κλεοπάτρα.
- Αυτή η διαδικασία μπορεί να κληθεί **το πολύ 2100 φορές** σε κάθε testcase.

Αυτή η διαδικασία πρέπει να καλέσει την παρακάτω διαδικασία για να στείλει ένα πακέτο:

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

- A: ένα αρχικό πακέτο (ένας πίνακας μήκους 31) που αντιπροσωπεύει τα bits που στέλνει η Αϊσά.
- Αυτή η διαδικασία επιστρέφει ένα μολυσμένο πακέτο B που αντιπροσωπεύει τα bits που θα λάβει η Μπάσμα.
- Αυτή η διαδικασία μπορεί να κληθεί το πολύ 100 φορές σε κάθε κλήση του `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 τα πακέτα που έστειλε η Αϊσά σε μία κλήση του `send_message` και δίνονται **με τη σειρά που στάλθηκαν** από την Αϊσά. Κάθε στοιχείο του R είναι ένας πίνακας μήκους 31, που αντιπροσωπεύει ένα μολυσμένο πακέτο.
- Αυτή η διαδικασία πρέπει να επιστρέψει έναν πίνακα από S bits που είναι ίσος με το αρχικό μήνυμα M.
- Αυτή η διαδικασία μπορεί να κληθεί **πολλές φορές** σε κάθε testcase, **ακριβώς μία φορά** για κάθε αντίστοιχη κλήση του `send_message`. Η **σειρά των** κλήσεων της διαδικασίας `receive_message` δεν είναι απαραίτητα η ίδια με τη σειρά των αντίστοιχων κλήσεων του `send_message`.

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

Περιορισμοί

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

Υποπροβλήματα και Βαθμολόγηση

Αν σε οποιαδήποτε από τα testcases, οι κλήσεις της διαδικασίας `send_packet` δεν συμμορφώνονται με τους κανόνες που αναφέρονται παραπάνω, ή η τιμή επιστροφής οποιασδήποτε από τις κλήσεις της διαδικασίας `receive_message` είναι λανθασμένη, η βαθμολογία της λύσης σας για αυτό το testacase θα είναι 0.

Διαφορετικά, έστω Q ο μέγιστος αριθμός κλήσεων της διαδικασίας `send_packet` μεταξύ όλων των κλήσεων του `send_message` σε όλα τα testcases. Επίσης, έστω X ίσο με:

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

Τότε, η βαθμολογία υπολογίζεται ως εξής:

Υποκατηγορία	Βαθμολογία	Πρόσθετοι Περιορισμοί
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Χωρίς πρόσθετους περιορισμούς.

Σημειώστε ότι σε ορισμένες περιπτώσεις η συμπεριφορά του 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 πολλά άλλα πράγματα, συμπεριλαμβανομένου τις εισόδους και τις τιμές επιστροφής των προηγούμενων κλήσεων σε αυτή τη διαδικασία και ψευδοτυχαίους αριθμούς που δημιουργούνται από τον grader. Ο grader είναι **ντετερμινιστικός** με την έννοια ότι αν τον τρέξεις δύο φορές, με τα ίδια πακέτα, θα κάνει τις ίδιες αλλαγές σε αυτά.

Παράδειγμα

Ας εξετάσουμε την παρακάτω 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 η Αϊσά προσπαθεί να στείλει στην Μπάσμα είναι $[0, 1, 1, 0]$. Τα bits στις θέσεις από το 0 έως το 15 δεν μπορούν να αλλαχθούν από την Κλεοπάτρα, ενώ τα bits στις θέσεις από το 16 έως το 30 μπορούν να αλλαχθούν από την Κλεοπάτρα.

Για χάρη του παραδείγματος, ας υποθέσουμε ότι η Κλεοπάτρα γεμίζει διαδοχικά bits που ελέγχει με εναλλασσόμενα 0 και 1, δηλαδή αποδίδει 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 Αϊσά μπορεί να αποφασίσει να στείλει δύο bits από το αρχικό μήνυμα σε ένα πακέτο ως εξής: θα στείλει το πρώτο bit στις πρώτες 8 θέσεις που ελέγχει και το δεύτερο bit στις επόμενες 8 θέσεις που ελέγχει.

Η Αϊσά στη συνέχεια επιλέγει να στείλει το παρακάτω 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 τα bits στις τελευταίες 15 θέσεις, οπότε η Αϊσά μπορεί να τα ορίσει αυθαίρετα, καθώς ενδέχεται να αντικατασταθούν. Με την υποτιθέμενη στρατηγική της Κλεοπάτρας, η διαδικασία επιστρέφει: $[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]$.

Η Αϊσά αποφασίζει να στείλει τα τελευταία δύο 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: $[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]$.

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

Ο βαθμολογητής στη συνέχεια πραγματοποιεί την ακόλουθη κλήση διαδικασίας:

```
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 Μπάσμα 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 $[1, 0]$. Βάζοντας τα μαζί, ανακτά το μήνυμα $[0, 1, 1, 0]$, που είναι η σωστή τιμή επιστροφής για αυτή την κλήση της `receive_message`.

Μπορεί να αποδειχθεί ότι με την υποτιθέμενη στρατηγική της Κλεοπάτρας και για μηνύματα μήκους 4, αυτή η προσέγγιση της Μπάσμα ανακτά σωστά το M, ανεξαρτήτως της τιμής του C. Ωστόσο, δεν είναι σωστή στη γενική περίπτωση.

Υπόδειγμα Grader

Ο δοσμένος grader δεν είναι προσαρμοστικός. Πιο συγκεκριμένα, η Κλεοπάτρα γεμίζει διαδοχικά bits που ελέγχει με εναλλασσόμενα 0 και 1 bits, όπως περιγράφεται στο παραπάνω παράδειγμα.

Μορφή εισόδου: **Η πρώτη γραμμή της εισόδου περιέχει έναν ακέραιο T, που καθορίζει τον αριθμό των σεναρίων.** Ακολουθούν T σενάρια. Το καθένα παρέχεται στην ακόλουθη μορφή:

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

Μορφή εξόδου: Ο δοσμένος grader γράφει το αποτέλεσμα κάθε ενός από τα T σενάρια με την ίδια σειρά που δίνονται στην είσοδο στην ακόλουθη μορφή:

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

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