Μυστικός Δείπνος

Ο Λεονάρντο εργαζόταν πολύ σκληρά όταν ζωγράφιζε το Μυστικό Δείπνο, την πιο διάσημη από τις τοιχογραφίες του: μία από τις πρώτες του δουλειές κάθε μέρα ήταν να αποφασίσει ποια χρώματα θα χρησιμοποιούσε την υπόλοιπη μέρα. Χρειαζόταν πολλά χρώματα, μπορούσε όμως να έχει μόνο έναν περιορισμένο αριθμό χρωμάτων στη σκαλωσιά του. Ο βοηθός του, μεταξύ άλλων, είχε την υποχρέωση να ανεβοκατεβαίνει στη σκαλωσιά δίνοντάς του χρώματα και παίρνοντας άλλα, τα οποία επέστρεφε στο ράφι του εργαστηρίου.

Σε αυτό το πρόβλημα, θα χρειαστεί να γράψετε δύο ξεχωριστά προγράμματα που θα χρησιμοποιήσει ο βοηθός. Το πρώτο πρόγραμμα θα δέχεται τις εντολές του Λεονάρντο (μία ακολουθία από χρώματα που ο Λεονάρντο θα ζητήσει κατά τη διάρκεια της ημέρας) και θα παράγει ένα *μικρό* σχέδιο για το βοηθό, που ονομάζεται *συμβουλή*. Κατά τη διάρκεια της ημέρας, ο βοηθός δε θα έχει πρόσβαση στα αιτήματα του Λεονάρντο, μόνο στη συμβουλή που παρήγαγε το πρώτο σας πρόγραμμα. Το δεύτερο πρόγραμμα θα δέχεται τη συμβουλή, και στη συνέχεια θα δέχεται και θα επεξεργάζεται τα αιτήματα του Λεονάρντο ένα τη φορά. Το πρόγραμμα πρέπει να μπορεί να καταλάβει τι σημαίνει η συμβουλή και να τη χρησιμοποιήσει για να κάνει τη βέλτιστη επιλογή. Όλα αυτά εξηγούνται λεπτομερώς παρακάτω.

Μετακίνηση χρωμάτων μεταξύ ραφιού και σκαλωσιάς

Έστω το εξής απλοποιημένο σενάριο. Ας υποθέσουμε ότι υπάρχουν N χρώματα, αριθμημένα από 0 έως N - 1, και ότι κάθε μέρα ο Λεονάρντο ζητάει από το βοηθό του ακριβώς N φορές κάποιο καινούργιο χρώμα. Έστω C η ακολουθία των N χρωμάτων που ζητάει ο Λεονάρντο. Έτσι, μπορούμε να σκεφτόμαστε το C ως μια ακολουθία N αριθμών, καθένας από τους οποίους είναι μεταξύ 0 και N - 1, συμπεριλαμβανομένων. Προσέξτε ότι κάποια χρώματα μπορεί να μην εμφανίζονται καθόλου στην C και κάποια άλλα μπορεί να εμφανίζονται περισσότερες φορές.

Η σκαλωσιά είναι πάντοτε γεμάτη και περιέχει K από τα N χρώματα, με K < N. Αρχικά, η σκαλωσιά περιέχει τα χρώματα από το 0 έως το K - 1, συμπεριλαμβανομένων.

Ο βοηθός επεξεργάζεται τα αιτήματα του Λεονάρντο ένα τη φορά. Όταν το χρώμα που ζητάει είναι *ήδη στη σκαλωσιά*, ο βοηθός ξεκουράζεται. Διαφορετικά, πρέπει να βρει το σωστό χρώμα στο ράφι και να το πάει στη σκαλωσιά. Φυσικά, στη σκαλωσιά δεν υπάρχει χώρος για να το προσθέσει, έτσι ο βοηθός είναι υποχρεωμένος να επιλέξει ένα από τα χρώματα που βρίσκει εκεί και να το πάρει μαζί του πίσω στο ράφι.

Η βέλτιστη στρατηγική του Λεονάρντο

Ο βοηθός θέλει να ξεκουραστεί όσο περισσότερες φορές είναι δυνατόν. Το πλήθος των αιτημάτων για τα οποία μπορεί να ξεκουραστεί εξαρτάται από τις επιλογές που κάνει, κατά τη

διάρκεια αυτής της διαδικασίας. Πιο συγκεκριμένα, καθε φορά που ο βοηθός πρέπει να αφαιρέσει ένα χρώμα από τη σκαλωσιά, διαφορετικές επιλογές του μπορεί να οδηγήσουν σε διαφορετικά αποτελέσματα στο μέλλον. Ο Λεονάρντο του εξηγεί πώς μπορεί να πετύχει το σκοπό του αν γνωρίζει το C. Η καλύτερη επιλογή για να αφαιρεθεί ένα χρώμα από τη σκαλωσιά επιτυγχάνεται κοιτάζοντας τα χρώματα που είναι τώρα στη σκαλωσιά, και τα υπόλοιπα αιτήματα στο C. Το χρώμα μεταξύ αυτών που είναι στη σκαλωσιά που πρέπει να αφαιρεθεί επιλέγεται ακολουθώντας τους εξής κανόνες:

- Αν υπάρχει ένα χρώμα στη σκαλωσιά που δε θα ξαναχρειαστεί στο μέλλον, ο βοηθός πρέπει να αφαιρέσει αυτό από τη σκαλωσιά.
- Διαφορετικά, το χρώμα που θα αφαιρεθεί από τη σκαλωσιά πρέπει να είναι αυτό *που θα χρειαστεί ξανά όσο γίνεται πιο αργά στο μέλλον*. (Δηλαδή, για κάθε ένα από τα χρώματα στη σκαλωσιά βρίσκουμε την πρώτη του μελλοντική εμφάνιση. Το χρώμα που επιστρέφεται στο ράφι είναι αυτό που θα χρειαστεί τελευταίο.) Μπορεί να αποδειχθεί ότι, χρησιμοποιώντας τη στρατηγική του Λεονάρντο, ο βοηθός θα ξεκουραστεί όσο περισσότερες φορές είναι δυνατόν.

Παράδειγμα 1

Έστω N = 4, οπότε έχουμε 4 χρώματα (αριθμημένα από 0 έως 3) και 4 αιτήματα. Έστω ότι η ακολουθία των αιτημάτων είναι C = (2, 0, 3, 0). Επίσης, ας υποθέσουμε ότι K = 2. Δηλαδή, η σκαλωσιά του Λεονάρντο χωράει 2 χρώματα κάθε στιγμή. Όπως είπαμε παραπάνω, η σκαλωσιά αρχικά έχει τα χρώματα 0 και 1. Θα γράφουμε τα περιεχόμενα της σκαλωσιάς ως εξής: [0, 1]. Ένας δυνατός τρόπος για να χειριστεί ο βοηθός τα αιτήματα είναι ο εξής.

- Το πρώτο χρώμα που ζητείται (αριθμός 2) δεν είναι στη σκαλωσιά. Ο βοηθός το βάζει και αποφασίζει να βγάλει το χρώμα 1 από τη σκαλωσιά. Η σκαλωσιά τώρα έχει [0, 2].

- Το επόμενο χρώμα που ζητείται (αριθμός 0) είναι ήδη στη σκαλωσιά, οπότε ο βοηθός ξεκουράζεται.

- Για το τρίτο αίτημα (αριθμός 3), ο βοηθός αφαιρεί το χρώμα 0, αλλάζοντας τη σκαλωσιά σε [3, 2].

- Τέλος, το τελευταίο χρώμα που ζητείται (αριθμός 0) πρέπει να μεταφερθεί από το ράφι. Ο βοηθός αποφασίζει να αφαιρέσει το χρώμα 2 και η σκαλωσιά τώρα γίνεται [3, 0].

Προσέξτε ότι στο παραπάνω παράδειγμα ο βοηθός δεν ακολούθησε τη βέλτιστη στρατηγική του Λεονάρντο. Η βέλτιστη στρατηγική θα αφαιρούσε το χρώμα 2 στο τρίτο βήμα, και άρα ο βοηθός θα μπορούσε να ξεκουραστεί στο τελευταίο βήμα.

Η στρατητική του βοηθού όταν η μνήμη είναι περιορισμένη

Το πρωί, ο βοηθός ζητάει από το Λεονάρντο να γράψει το C σε ένα κομμάτι χαρτί, έτσι ώστε να μπορεί να εφαρμόσει τη βέλτιστη στρατηγική. Όμως, ο Λεονάρντο έχει την εμμονή να κρατάει κρυφές τις τεχνικές του στη ζωγραφική, έτσι αρνείται να δώσει το χαρτί στο βοηθό. Τον αφήνει μόνο να διαβάσει το C και να προσπαθήσει να το απομνημονεύσει.

Δυστυχώς, η μνήμη του βοηθού είναι πολύ κακή. Μπορεί να θυμάται μόνο μέχρι M bits. Γενικά, αυτό μπορεί να τον εμποδίσει να ανακατασκευάσει ολόκληρη την ακολουθία C. Έτσι, ο βοηθός

πρέπει να σκεφτεί κάποιον έξυπνο τρόπο για να υπολογίσει την ακολουθία των bits που θα θυμάται. Ονομάζουμε αυτή την ακολουθία *ακολουθία συμβουλών* και την παριστάνουμε με A.

Παράδειγμα 2

Το πρωί, ο βοηθός παίρνει το χαρτί του Λεονάρντο με την ακολουθία C, τη διαβάζει, και κάνει κάποιες επιλογές. Ένα από τα πράγματα που μπορεί να κάνει είναι να εξετάσει την κατάσταση της σκαλωσιάς στο τέλος κάθε αιτήματος. Για παράδειγμα, αν χρησιμοποιήσει την (υποβέλτιστη) στρατηγική που περιγράφεται στο Παράδειγμα 1, η ακολουθία καταστάσεων της σκαλωσιάς θα είναι [0, 2], [0, 2], [3, 2], [3, 0]. (Θυμηθείτε ότι ξέρει πως η αρχική κατάσταση της σκαλωσίας είναι πάντα [0, 1].)

Τώρα υποθέστε ότι M = 16, επομένως ο βοηθός μπορεί να θυμάται μέχρι 16 bits πληροφορίας. Καθώς είναι N = 4, μπορούμε να αποθηκεύσουμε κάθε χρώμα χρησιμοποιώντας 2 bits. Επομένως 16 bits αρκούν για να αποθηκεύσουμε ολόκληρη την ακολουθία καταστάσεων της σκαλωσιάς. Άρα, ο βοηθός υπολογίζει την εξής ακολουθία συμβουλών: A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`).

Αργότερα, ο βοηθός θα μπορεί να αποκωδικοποιήσει αυτή την ακολουθία συμβουλών και να τη χρησιμοποιήσει για να κάνει τις επιλογές του.

(Φυσικά, με M = 16 ο βοηθός μπορεί, αντί να κάνει αυτό, να απομνημονεύσει ολόκληρη την ακολουθία C, χρησιμοποιώντας μόνο 8 από τα διαθέσιμα 16 bits. Σε αυτό το παράδειγμα θέλαμε απλώς να δείξουμε ότι μπορεί να υπάρχουν και άλλες επιλογές, χωρίς να προσπαθούμε να υποδείξουμε κάποια καλή λύση.)

Πρόβλημα

Ζητείται να γράψετε *δύο διαφορετικά προγράμματα* στην ίδια γλώσσα προγραμματισμού. Τα προγράμματα αυτά θα εκτελούνται το ένα μετά το άλλο, χωρίς να μπορούν να επικοινωνούν μεταξύ τους κατά τη διάρκεια της εκτέλεσής τους.

Το πρώτο πρόγραμμα θα είναι αυτό που θα χρησιμοποιεί ο βοηθός το πρωί. Το πρόγραμμα αυτό θα δέχεται την ακολουθία C και θα πρέπει να υπολογίσει την ακολουθία συμβουλών A.

Το δεύτερο πρόγραμμα θα είναι αυτό που θα χρησιμοποιεί ο βοηθός την υπόλοιπη ημέρα. Το πρόγραμμα αυτό θα δέχεται την ακολουθία συμβουλών A και θα επεξεργάζεται την ακολουθία C των αιτημάτων του Λεονάρντο. Προσέξτε ότι η ακολουθία C θα αποκαλύπτεται στο πρόγραμμα ένα αίτημα τη φορά και ότι κάθε αίτημα θα πρέπει να ικανοποιηθεί πριν γίνει γνωστό το επόμενο.

Ακριβέστερα, στο πρώτο πρόγραμμα πρέπει να υλοποιήσετε μία ρουτίνα `ComputeAdvice`(C, N, K, M) που έχει ως είσοδο έναν πίνακα C από N ακέραιους (κάθε ένας μεταξύ 0, …, N - 1), το πλήθος K των χρωμάτων στη σκαλωσιά, και το πλήθος M των bits που είναι διαθέσιμα για την ακολουθία συμβουλών. Το πρόγραμμα πρέπει να υπολογίζει την ακολουθία συμβουλών A που αποτελείται από το πολύ M bits. Το πρόγραμμα πρέπει στη συνέχεια να επικοινωνεί την ακολουθία συμβουλών A στο σύστημα, καλώντας για κάθε bit της A με τη σειρά την ακόλουθη ρουτίνα:

- `WriteAdvice`(B) — προσθέτει το bit B στην τρέχουσα ακολουθία συμβουλών A. (Μπορείτε να καλέσετε αυτή τη ρουτίνα το πολύ M φορές.)

Στο δεύτερο πρόγραμμα πρέπει να υλοποιήσετε μία ρουτίνα `Assist`(A, N, K, R). Η είσοδος αυτής της ρουτίνας είναι η ακολουθία συμβουλών A, οι ακέραιοι N και K όπως ορίστηκαν παραπάνω, και το πραγματικό μήκος R της ακολουθίας συμβουλών A σε bits ($R \leq M$). Η ρουτίνα αυτή πρέπει να υλοποιεί τη στρατηγική που προτείνετε να ακολουθήσει ο βοηθός, χρησιμοποιώντας τις ακόλουθες ρουτίνες που σας δίνονται:

- `GetRequest`() — επιστρέφει το επόμενο χρώμα που ζητάει ο Λεονάρντο. (Δεν αποκαλύπτεται καμία πληροφορία σχετικά με μελλοντικά αιτήματα.)

- `PutBack`(T) — επιστρέφει το χρώμα T από τη σκαλωσιά πίσω στο ράφι. Μπορείτε να καλέσετε αυτή τη ρουτίνα με ένα χρώμα T που υπάρχει εκείνη τη στιγμή στη σκαλωσιά.

Όταν εκτελείται, η ρουτίνα σας `Assist` πρέπει να καλεί τη `GetRequest` ακριβώς N φορές, κάθε φορά λαμβάνοντας ένα αίτημα του Λεονάρντο, με τη σειρά. Μετά από κάθε κλήση στην `GetRequest`, αν το χρώμα που επιστρέφεται *δεν* είναι στη σκαλωσιά, *πρέπει* να καλέσετε επίσης την `PutBack`(T) με την επιλογή του T που κάνετε. Διαφορετικά *δεν πρέπει* να καλέσετε την `PutBack`. Αν αποτύχετε να κάνετε τα παραπάνω, θα θεωρηθεί λάθος και θα προκαλέσει τον τερματισμό του προγράμματός σας. Θυμηθείτε επίσης ότι αρχικά η σκαλωσιά περιέχει τα χρώματα από 0 έως K - 1, συμπεριλαμβανομένων.

Μία συγκεκριμένη περίπτωση ελέγχου θα θεωρείται επιτυχής αν οι ρουτίνες σας ακολουθούν τους περιορισμούς που έχουν τεθεί και το συνολικό πλήθος κλήσεων στην `PutBack` είναι *ακριβώς ίσο* με αυτό της βέλτιστης στρατηγικής του Λεονάρντο. Προσέξτε ότι αν υπάρχουν περισσότερες στρατηγικές που επιτυγχάνουν το ίδιο πλήθος κλήσεων της `PutBack`, το πρόγραμμά σας είναι ελεύθερο να υλοποιεί οποιαδήποτε θέλει. (Δηλαδή, δεν είναι υποχρεωτικό να ακολουθεί τη στρατηγική του Λεονάρντο, αν υπάρχει και άλλη εξίσου καλή στρατηγική.)

Παράδειγμα 3

Συνεχίζοντας το Παράδειγμα 2, ας υποθέσουμε ότι στην `ComputeAdvice` υπολογίσατε A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`). Για να επικοινωνήσετε αυτή την ακολουθία προς το σύστημα, πρέπει να κάνετε τις εξής κλήσεις κατά σειρά: `WriteAdvice`(`0`), `WriteAdvice`(`0`), `WriteAdvice`(`1`), `WriteAdvice`(`0`), `WriteAdvice`(`0`), `WriteAdvice`(`0`), `WriteAdvice`(`1`), `WriteAdvice`(`0`), `WriteAdvice`(`1`), `WriteAdvice`(`1`), `WriteAdvice`(`0`), `WriteAdvice`(`1`), `WriteAdvice`(`1`), `WriteAdvice`(`0`), `WriteAdvice`(`0`).

Η δεύτερη ρουτίνα σας `Assist` θα εκτελεστεί τότε, λαμβάνοντας την παραπάνω ακολουθία συμβουλών A, και τις τιμές N = 4, K = 2, και R = 16. Η ρουτίνα `Assist` πρέπει να εκτελεί ακριβώς N = 4 κλήσεις στην `GetRequest`. Επίσης, μετά από ορισμένα από τα αιτήματα, η `Assist` πρέπει να καλεί την `PutBack`(T) με κάποιο κατάλληλο χρώμα T.

Ο παρακάτω πίνακας δείχνει μία ακολουθία κλήσεων που αντιστοιχεί στις (υποβέλτιστες) επιλογές του Παραδείγματος 1. Η παύλα σημαίνει ότι δεν έγινε κλήση στην `PutBack`.

`GetRequest()`	Ενέργεια
2	`PutBack`(1)
0	-
3	`PutBack`(0)
0	`PutBack`(2)

Υποπρόβλημα 1 [8 βαθμοί]

- $N \leq 5\,000$.
- Μπορείτε να χρησιμοποιήσετε το πολύ M = 65 000 bits.

Υποπρόβλημα 2 [9 βαθμοί]

- $N \leq 100\,000$.
- Μπορείτε να χρησιμοποιήσετε το πολύ M = 2 000 000 bits.

Υποπρόβλημα 3 [9 βαθμοί]

- $N \leq 100\,000$.
- $K \leq 25\,000$.
- Μπορείτε να χρησιμοποιήσετε το πολύ M = 1 500 000 bits.

Υποπρόβλημα 4 [35 βαθμοί]

- $N \leq 5\,000$.
- Μπορείτε να χρησιμοποιήσετε το πολύ M = 10 000 bits.

Υποπρόβλημα 5 [μέχρι 39 βαθμοί]

- $N \leq 100\,000$.
- $K \leq 25\,000$.
- Μπορείτε να χρησιμοποιήσετε το πολύ M = 1 800 000 bits.

Η βαθμολογία για αυτό το υποπρόβλημα εξαρτάται από το μήκος R της ακολουθίας συμβουλών που επικοινωνεί το πρόγραμμά σας. Ακριβέστερα, αν R_{max} είναι το μέγιστο (από όλες τις περιπτώσεις ελέγχου) μήκος της ακολουθίας συμβουλών που παράγει η ρουτίνα σας `ComputeAdvice`, η βαθμολογία σας θα είναι:

- 39 βαθμοί αν $R_{max} \leq 200\,000$,
- 39 $(1\,800\,000 - R_{max}) / 1\,600\,000$ βαθμοί αν $200\,000 < R_{max} < 1\,800\,000$,

- 0 βαθμοί αν $R_{max} \geq 1\ 800\ 000$.

Λεπτομέρειες υλοποίησης

Πρέπει να υποβάλετε ακριβώς δύο αρχεία *στην ίδια γλώσσα προγραμματισμού*.

Το πρώτο αρχείο λέγεται `advisor.c`, `advisor.cpp` ή `advisor.pas`. Το αρχείο αυτό πρέπει να υλοποιεί τη ρουτίνα `ComputeAdvice` όπως περιγράφεται παραπάνω και μπορεί να καλεί τη ρουτίνα `WriteAdvice`. Το δεύτερο αρχείο λέγεται `assistant.c`, `assistant.cpp` ή `assistant.pas`. Το αρχείο αυτό πρέπει να υλοποιεί τη ρουτίνα `Assist` όπως περιγράφεται παραπάνω και μπορεί να καλεί τις ρουτίνες `GetRequest` και `PutBack`.

Ακολουθούν οι επικεφαλίδες όλων των ρουτίνων.

Προγράμματα C/C++

```
void ComputeAdvice(int *C, int N, int K, int M);
void WriteAdvice(unsigned char a);
```

```
void Assist(unsigned char *A, int N, int K, int R);
void PutBack(int T);
int GetRequest();
```

Προγράμματα Pascal

```
procedure ComputeAdvice(var C : array of LongInt; N, K, M : LongInt);
procedure WriteAdvice(a : Byte);
```

```
procedure Assist(var A : array of Byte; N, K, R : LongInt);
procedure PutBack(T : LongInt);
function GetRequest : LongInt;
```

Οι ρουτίνες αυτές πρέπει να συμπεριφέρονται ακριβώς όπως περιγράφεται παραπάνω. Φυσικά είστε ελεύθεροι να υλοποιήσετε και άλλες ρουτίνες για εσωτερική χρήση. Για προγράμματα C/C++, οι εσωτερικές σας ρουτίνες πρέπει να δηλώνονται ως `static`, καθώς ο ενδεικτικός διορθωτής θα συνδέσει τα προγράμματα μαζί. Εναλλακτικά, απλώς αποφύγετε να έχετε δύο ρουτίνες (μία σε κάθε πρόγραμμα) που να έχουν το ίδιο όνομα. Οι υποβολές σας δεν πρέπει να επικοινωνούν με κανέναν τρόπο με το standard input/output, ούτε με κανένα άλλο αρχείο.

Όταν προγραμματίζετε τη λύση σας, πρέπει επίσης να ακολουθήσετε τις παρακάτω οδηγίες (τα δείγματα προγραμμάτων που θα βρείτε στο περιβάλλον διαγωνισμού ήδη ικανοποιούν τους παρακάτω περιορισμούς).

Προγράμματα C/C++

Στην αρχή της λύσης σας, πρέπει να εισαγάγετε αρχεία `advisor.h` και `assistant.h`, αντίστοιχα, στα προγράμματα advisor και assistant. Αυτό γίνεται βάζοντας στον κώδικά σας τη γραμμή:

```
#include "advisor.h"
```

ή

```
#include "assistant.h"
```

Τα δύο αρχεία `advisor.h` και `assistant.h` σας δίνονται σε ένα directory μέσα στο περιβάλλον διαγωνισμού και δίνονται επίσης στο Web interface του διαγωνισμού. Θα σας δοθεί επίσης (με τον ίδιο τρόπο) κώδικας και scripts για να μεταγλωττίζετε και να ελέγχετε τις λύσεις σας. Συγκεκριμένα, αφού αντιγράψετε τη λύση σας στο directory που βρίσκονται αυτά τα scripts, θα πρέπει να τρέξετε `compile_c.sh` ή `compile_cpp.sh` (ανάλογα με την επιλογή γλώσσας σας).

Προγράμματα Pascal

Πρέπει να χρησιμοποιήσετε τις μονάδες `advisorlib` και `assistantlib`, αντίστοιχα, στα προγράμματα advisor και assistant. Αυτό γίνεται βάζοντας στον κώδικά σας τη γραμμή:

```
uses advisorlib;
```

ή

```
uses assistantlib;
```

Τα δύο αρχεία `advisorlib.pas` και `assistantlib.pas` σας δίνονται σε ένα directory μέσα στο περιβάλλον διαγωνισμού και δίνονται επίσης στο Web interface του διαγωνισμού. Θα σας δοθεί επίσης (με τον ίδιο τρόπο) κώδικας και scripts για να μεταγλωττίζετε και να ελέγχετε τις λύσεις σας. Συγκεκριμένα, αφού αντιγράψετε τη λύση σας στο directory που βρίσκονται αυτά τα scripts, θα πρέπει να τρέξετε `compile_pas.sh`.

Ενδεικτικός διορθωτής

Ο ενδεικτικός διορθωτής θα δέχεται την είσοδό του με την εξής μορφή:

- γραμμή 1: N, K, M;
- γραμμές 2, …, N + 1: C[i].

Ο διορθωτής θα εκτελεί πρώτα τη ρουτίνα `ComputeAdvice`. Αυτό θα παράγει ένα αρχείο `advice.txt`, που θα περιέχει τα bits της ακολουθίας συμβουλών, χωρισμένα με κενά διαστήματα και τελειώνοντας με ένα 2.

Στη συνέχεια θα εκτελεί τη ρουτίνα σας `Assist` και θα παράγει έξοδο στην οποία κάθε γραμμή είναι είτε της μορφής "`R [number]`", ή της μορφής "`P [number]`". Οι γραμμές της πρώτης μορφής δείχνουν κλήσεις στην `GetRequest()` και τις αντίστοιχες απαντήσεις. Οι γραμμές της δεύτερης μορφής δείχνουν κλήσεις στην `PutBack`() και τα χρώματα που επιλέγονται για επιστροφή. Η έξοδος τερματίζεται με μία γραμμή της μορφής "`E`".

Προσέξτε ότι στον επίσημο διορθωτή ο χρόνος εκτέλεσης του προγράμματος μπορεί να διαφέρει λίγο από το χρόνο εκτέλεσης στον τοπικό υπολογιστή. Η διαφορά δεν πρέπει να είναι σημαντική. Σε κάθε περίπτωση, σας συνιστούμε να χρησιμοποιείτε το test interface για να διαπιστώνετε αν η λύση σας εκτελείται μέσα στο όριο χρόνου.

Περιορισμοί χρόνου και μνήμης

- Όριο χρόνου: 7 δευτερόλεπτα.
- Όριο μνήμης: 256 MiB.