37ος
ΠΑΝΕΛΛΗΝΙΟΣ ΔΙΑΓΩΝΙΣΜΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΘΕΜΑΤΑ Α΄ ΦΑΣΗΣ

Φέτος, για πρώτη φορά, δεν υπάρχει μόνο ένα θέμα στην Α' φάση του ΠΔΠ, αλλά τέσσερα! Η δυσκολία των περισσότερων προβλημάτων δεν είναι διαφορετική, σε σχέση με τις προηγούμενες χρονιές, η μορφή τους όμως είναι κάπως διαφορετική σε ορισμένα σημεία. Διαβάστε με προσοχή τα παρακάτω και τις εκφωνήσεις!

1. Κάθε πρόβλημα αποτελείται από περισσότερα **υποπροβλήματα** (στο στυλ της IOI), καθένα από τα οποία σας δίνει κάποιους βαθμούς, αν το λύσετε σωστά. Τα υποπροβλήματα περιέχουν περιορισμούς που κάνουν το πρόβλημα ευκολότερο από τη γενική του μορφή. Π.χ., στο πρώτο πρόβλημα (**samepizzas**), το πρώτο υποπρόβλημα λέει ότι τα διαθέσιμα υλικά για κάθε πίτσα είναι μόνο δύο, το δεύτερο υποπρόβλημα λέει ότι κάθε πίτσα πρέπει να περιέχει όλα τα υλικά, κ.ο.κ.
2. Μπορείτε να λύσετε ανεξάρτητα όποια υποπροβλήματα θέλετε, παίρνοντας τους αντίστοιχους βαθμούς. Όμως, για να πάρετε τους βαθμούς ενός υποπροβλήματος πρέπει η λύση σας να απαντάει σωστά και εντός ορίων χρόνου και μνήμης σε **όλες** τις περιπτώσεις ελέγχου του υποπροβλήματος. Δείτε τις σχετικές οδηγίες στο Hellenico για το πώς να οργανώσετε κατάλληλα τις υποβολές σας.
3. Για να προκριθείτε στην επόμενη φάση **δε χρειάζεται να λύσετε όλα** τα προβλήματα! Αρκεί να συγκεντρώσετε αρκετούς βαθμούς, λύνοντας όσα περισσότερα υποπροβλήματα μπορείτε. Εκτιμούμε ότι 50-100 βαθμοί θα σας είναι αρκετοί για να περάσετε στην επόμενη φάση!
4. Το τρίτο πρόβλημα (**finding**) είναι **διαδραστικό** (interactive). Διαβάστε προσεκτικά την εκφώνησή του και παρακολουθείτε την ιστοσελίδα του ΠΔΠ και το Facebook group για υποδείξεις.
5. Φροντίστε να διαβάζετε την είσοδο και να εκτυπώνετε την έξοδο αποδοτικά, στα προβλήματα που το απαιτούν.
6. Σε κάποια προβλήματα η απάντηση μπορεί να υπερβαίνει το 2^{32} και εν γένει μπορεί να χρειαστεί να κάνετε πράξεις με αριθμούς μεγαλύτερους από αυτούς που μπορούν να αναπαρασταθούν με τον βασικό τύπο **int** ή **integer**, της γλώσσας προγραμματισμού που χρησιμοποιείτε.

7. **Επικεφαλίδες στον πηγαίο κώδικα**: Στην αρχή κάθε αρχείου πηγαίου κώδικα που υποβάλλετε στο Hellenico, θα πρέπει να χρησιμοποιήσετε τις παρακάτω επικεφαλίδες.
   - Στη θέση του ονόματος χρήστη (username) συμπληρώστε το δικό σας όνομα χρήστη στο Hellenico.
   - Στη θέση του ονόματος του προβλήματος (taskname), συμπληρώστε το αντίστοιχο όνομα από την εκφώνηση (π.χ. samepizzas για το πρώτο πρόβλημα).

```
(* USER: username
LANG: PASCAL
TASK: taskname *)
```
για κώδικα σε PASCAL

```
/* USER: username
LANG: C
TASK: taskname */
```
για κώδικα σε C

```
/* USER: username
LANG: C++
TASK: taskname */
```
για κώδικα σε C++

```
/* USER: username
LANG: Java
TASK: taskname */
```
για κώδικα σε Java

Οι ιστορίες στις εκφωνήσεις των θεμάτων είναι προϊόντα μυθοπλασίας. Τα πρόσωπα, τα ονόματα και οι καταστάσεις είναι φανταστικά και οποιαδήποτε ομοιότητα με την πραγματικότητα είναι συμπτωματική.

Καλή επιτυχία!

ΟΜΟΙΕΣ ΠΙΤΣΕΣ

samepizzas

Σας παρουσιάζουμε τον πρωταγωνιστή μας τον Τάκη, έναν νεαρό Έλληνα που το όνειρό του από μικρό παιδί είναι να ανοίξει την δική του πιτσαρία. Σήμερα, βρίσκεται πολύ κοντά στο να κάνει αυτό το όνειρό του πραγματικότητα.

Για να εξοικονομήσει όσο γίνεται περισσότερα χρήματα, το πλάνο του Τάκη είναι να μην προσλάβει προσωπικό, αλλά να φτιάχνει και να σερβίρει ο ίδιος τις πίτσες του. Στην αποθήκη του διαθέτει **N** υλικά. Οι ποσότητες των διάφορων υλικών συμβολίζονται με τους θετικούς ακέραιους $\mathbf{a_1}$, $\mathbf{a_2}$, …, $\mathbf{a_N}$. Ο σκοπός του Τάκη είναι να δημιουργήσει όσο το δυνατόν περισσότερες όμοιες πίτσες μπορεί. Λέμε ότι κάποιες πίτσες είναι όμοιες όταν όλες:

- αποτελούνται από τα ίδια υλικά και
- διαθέτουν την ίδια (θετική ακέραια) ποσότητα από καθένα από αυτά τα υλικά.

Προκειμένου οι πίτσες του να μην είναι "βαρετές", ο Τάκης έθεσε ακόμα έναν περιορισμό: πρέπει κάθε πίτσα να περιέχει τουλάχιστον **K** διαφορετικά υλικά. Να υπολογίσετε ποιο είναι το μέγιστο πλήθος από όμοιες πίτσες που μπορεί να δημιουργήσει ο Τάκης, τηρώντας ταυτόχρονα αυτόν τον περιορισμό.

Πρόβλημα:

Να αναπτύξετε ένα πρόγραμμα σε μια από τις γλώσσες του ΠΔΠ (PASCAL, C, C++, Java) το οποίο θα διαβάζει τα δεδομένα εισόδου από το αρχείο **samepizzas.in** και θα εκτυπώνει τα αποτελέσματα στο αρχείο **samepizzas.out**.

Δεδομένα εισόδου (samepizzas.in):

Η πρώτη γραμμή περιέχει τον αριθμό του υποπροβλήματος (βλ. παρακάτω). Η δεύτερη γραμμή περιέχει δύο ακέραιους αριθμούς **N** και **K**, χωρισμένους μεταξύ τους με ένα κενό διάστημα: το πλήθος των υλικών που διαθέτει ο Τάκης και το ελάχιστο δυνατό πλήθος διαφορετικών υλικών που μπορεί να περιέχει κάθε πίτσα του. Η τρίτη

γραμμή αποτελείται από **N** ακέραιους αριθμούς $\mathbf{a_1}$, $\mathbf{a_2}$, …, $\mathbf{a_N}$, χωρισμένους ανά δύο με κενό διάστημα: τις ποσότητες των υλικών.

Δεδομένα εξόδου (samepizzas.out):

Πρέπει να περιέχει μία μόνο γραμμή με έναν ακέραιο αριθμό: το μέγιστο πλήθος από όμοιες πίτσες που μπορεί να δημιουργήσει ο Τάκης πληρώντας τον παραπάνω περιορισμό.

Παραδείγματα εισόδου - εξόδου:

samepizzas.in	samepizzas.out
4 5 3 12 6 7 6 4	6

Εξήγηση: Ο Τάκης μπορεί να φτιάξει 6 όμοιες πίτσες, που καθεμία περιέχει δύο μονάδες από το πρώτο, μία μονάδα από το δεύτερο και μία μονάδα από το τέταρτο υλικό. Μπορεί επίσης να κάνει και άλλους συνδυασμούς υλικών που δίνουν 6 όμοιες πίτσες, δεν μπορεί όμως με κανέναν τρόπο να φτιάξει περισσότερες.

Περιορισμοί:

- $1 \leq K \leq N \leq 200.000$
- $1 \leq a_i \leq 1.000.000.000$

Υποπροβλήματα:

1. (10 βαθμοί) N = 2
2. (15 βαθμοί) K = N
3. (25 βαθμοί) K = 1
4. (10 βαθμοί) $N \leq 10.000$
5. (40 βαθμοί) Δεν υπάρχουν περαιτέρω περιορισμοί

Παρατηρήσεις:

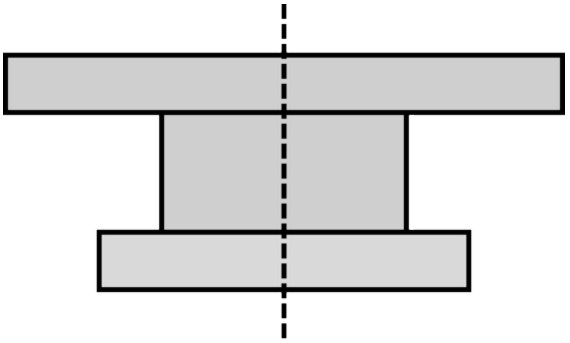
Μορφοποίηση: Στην είσοδο αλλά και στην έξοδο, κάθε γραμμή τερματίζει με έναν χαρακτήρα `newline`.

Μέγιστος χρόνος εκτέλεσης: 1 sec.
Μέγιστη διαθέσιμη μνήμη: 64 MB.

ΠΟΛΥΓΩΝΟ ΚΟΥΤΙΩΝ — polybox

Η προσπάθεια του Τάκη να φτιάξει όμοιες πίτσες απέτυχε παταγωδώς! Για να μπορέσει να ξεχαστεί, ο Τάκης αποφάσισε να παίξει με μερικά κουτιά πίτσας. Συγκεκριμένα, έχει αγοράσει **N** κουτιά σε σχήμα ορθογωνίου παραλληλογράμμου. Το i-οστό κουτί έχει πλάτος $\mathbf{w_i}$ και ύψος $\mathbf{h_i}$.

Αρχικά, ο Τάκης έβαλε το πρώτο κουτί στο πάτωμα, έπειτα έβαλε το δεύτερο κουτί από πάνω του, και ούτω καθεξής, δημιουργώντας έναν πύργο. Τα κουτιά τοποθετήθηκαν με προσανατολισμό τέτοιον ώστε οι πλευρές με μήκος $\mathbf{w_i}$ να είναι παράλληλες προς το έδαφος. Επιπλέον, επειδή ο Τάκης είναι τελειομανής, φρόντισε ο πύργος του να έχει κατακόρυφο άξονα συμμετρίας, όπως φαίνεται στο ακόλουθο σχέδιο.



Ο Τάκης σας ζητάει να υπολογίσετε την περίμετρο του σχήματος που δημιουργείται, καθώς όταν προσπάθησε να το κάνει μόνος του ζαλίστηκε από τα πολλά νούμερα.

Πρόβλημα:

Να αναπτύξετε ένα πρόγραμμα σε μια από τις γλώσσες του ΠΔΠ (PASCAL, C, C++, Java) το οποίο θα διαβάζει τα δεδομένα εισόδου από το αρχείο **polybox.in** και θα εκτυπώνει τα αποτελέσματα στο αρχείο **polybox.out**.

Δεδομένα εισόδου (polybox.in):

Η πρώτη γραμμή περιέχει τον αριθμό του υποπροβλήματος (βλ. παρακάτω). Η δεύτερη γραμμή περιέχει έναν ακέραιο αριθμό **N**: το πλήθος των κουτιών. Ακολουθούν **N** γραμμές, καθεμία από τις οποίες περιέχει δύο ακέραιους αριθμούς $\mathbf{w_i}$ και $\mathbf{h_i}$, χωρισμένους μεταξύ τους

με ένα κενό διάστημα: το πλάτος και το ύψος του i-οστού κουτιού. Τα κουτιά δίνονται με τη σειρά που τοποθετούνται στον πύργο του Τάκη, από κάτω προς τα πάνω.

Δεδομένα εξόδου (polybox.out):

Πρέπει να περιέχει μία μόνο γραμμή με έναν ακέραιο αριθμό: την περίμετρο του σχήματος που δημιουργήθηκε.

Παραδείγματα εισόδου - εξόδου:

polybox.in	polybox.out
3 3 6 1 4 2 9 1	30

Εξήγηση: Το παράδειγμα αντιστοιχεί στο παραπάνω σχέδιο. Το κάτω κουτί έχει πλάτος 6 και ύψος 1, το μεσαίο έχει πλάτος 4 και ύψος 2 και το επάνω έχει πλάτος 9 και ύψος 1. Μπορεί κανείς να υπολογίσει, και με την βοήθεια του σχεδίου, ότι η περίμετρος αυτού του σχήματος ισούται με 30.

Περιορισμοί:

- $1 \leq N \leq 100.000$
- $1 \leq w_i, h_i \leq 10^{40}$

Υποπροβλήματα:

1. (5 βαθμοί) $w_1 = w_2 = \ldots = w_N$ (όλα τα πλάτη είναι ίσα), $w_i \leq 10^9$ και $h_i \leq 10^9$
2. (25 βαθμοί) $N \leq 2$, $w_i \leq 10^9$ και $h_i \leq 10^9$
3. (60 βαθμοί) $w_i \leq 10^9$ και $h_i \leq 10^9$
4. (10 βαθμοί) Δεν υπάρχουν περαιτέρω περιορισμοί

Παρατηρήσεις:

Μορφοποίηση: Στην είσοδο αλλά και στην έξοδο, κάθε γραμμή τερματίζει με έναν χαρακτήρα `newline`.

Μέγιστος χρόνος εκτέλεσης: 2 sec.
Μέγιστη διαθέσιμη μνήμη: 128 MB.

ΨΑΧΝΟΝΤΑΣ ΤΟΥΣ ΑΝΤΑΓΩΝΙΣΤΕΣ finding

Η πόλη στην οποία μένει ο Τάκης αποτελείται από **N** σπίτια, χτισμένα κατά μήκος μιας ευθείας γραμμής και αριθμημένα από το 1 έως και το **N**, από τα αριστερά προς τα δεξιά. Δύο διαδοχικά σπίτια απέχουν μεταξύ τους 1 μονάδα απόστασης.

Στην ίδια πόλη, στα σπίτια **A** και **B**, όπου **A** < **B**, μένουν δύο ανταγωνιστές του Τάκη. Τις τοποθεσίες των **A** και **B** δεν τις γνωρίζει ο Τάκης, τις γνωρίζει όμως ο φίλος του ο Γιάννης.

Ο Γιάννης δεν θέλει ο φίλος του να μπλέκεται σε καυγάδες, αλλά έχει βαρεθεί την υπερβολική ησυχία της πόλης του. Για αυτό θέλει να μαρτυρήσει στον Τάκη τις κρυφές τοποθεσίες με… έμμεσο τρόπο. Ο Τάκης μπορεί να υποδείξει στον Γιάννη ένα σπίτι **X** και αυτός θα του απαντήσει με το άθροισμα των αποστάσεων του **X** από τα σπίτια **A** και **B**, δηλαδή το |**X**−**A**| + |**X**−**B**|. Ο Τάκης δεν έχει χρόνο για χάσιμο. Θέλει να βρει τα σπίτια των ανταγωνιστών του κάνοντας προς τον Γιάννη το πολύ **Q** ερωτήματα αυτού του είδους.

Πριν ο Τάκης αρχίσει να ρωτάει, ο Γιάννης του ανακοινώνει τον αριθμό των σπιτιών, δηλαδή το **N**. Επίσης, ανάλογα με τα κέφια του, ο Γιάννης μπορεί να του αποκαλύψει και την απόσταση **D** μεταξύ των σπιτιών **A** και **B** (που προφανώς ισούται με **B**−**A**), μπορεί όμως και να μη θέλει να του την αποκαλύψει.

Περιέργως, ο Τάκης κατάφερε να λύσει το πρόβλημα. Τώρα είναι η σειρά σας! Μπείτε στην θέση του και βρείτε τα σπίτια **A** και **B**, χωρίς να ξεπεράσετε το όριο των **Q** ερωτημάτων ανά στιγμιότυπο.

Πρόβλημα:

Το πρόβλημα αυτό είναι διαδραστικό (interactive). Αν δεν έχετε ξανασυναντήσει τέτοιου είδους πρόβλημα, συμβουλευτείτε τον οδηγό στην ιστοσελίδα του ΠΔΠ. Από την ίδια ιστοσελίδα μπορείτε επίσης να κατεβάσετε το συμπιεσμένο αρχείο **grader.zip** που περιέχει ένα πρότυπο πρόγραμμα για να σας βοηθήσει να καταλάβετε το πώς λειτουργεί η αλληλεπίδραση, καθώς και έναν υποδειγματικό βαθμολογητή, μαζί με εντολές για να κάνετε compile και να τρέξετε τον κώδικά σας στον δικό σας υπολογιστή.

Αλληλεπίδραση:

Γράψτε ένα πρόγραμμα με το οποίο ο Τάκης θα αλληλεπιδρά με τον Γιάννη και θα προσπαθεί να λύσει ένα ή περισσότερα στιγμιότυπα του παραπάνω προβλήματος, ανεξάρτητα μεταξύ τους, με ενδεχομένως διαφορετικά **N**, **A**, **B**, αλλά με το ίδιο **Q**. Ο Γιάννης είτε θα αποκαλύπτει την απόσταση **D** των δύο σπιτιών σε όλα τα στιγμιότυπα του προβλήματος, είτε δε θα την αποκαλύπτει σε κανένα.

Είναι απαραίτητο να συμπεριλάβετε στον κώδικα σας τη γραμμή:

- #include "findinglib.h" (σε C++ ή C)
- uses findinglib; (σε Pascal)
- findinglib grader = new findinglib(); (σε Java)

Με αυτόν τον τρόπο αποκτάτε πρόσβαση στις παρακάτω συναρτήσεις, μέσω των οποίων μπορείτε να αλληλεπιδράσετε με τον Γιάννη:

- int getSubtask();
- int getN();
- int getD();
- int query(int X);
- bool answer(int A, int B);

Οι συναρτήσεις αυτές λειτουργούν ως εξής:

1. Η συνάρτηση getSubtask() επιστρέφει τον αριθμό του υποπροβλήματος (βλ. παρακάτω), ο οποίος θα είναι ίδιος για όλα τα στιγμιότυπα μιας περίπτωσης ελέγχου.
2. Η συνάρτηση getN() επιστρέφει το πλήθος των σπιτιών, **N**.
3. Η συνάρτηση getD() επιστρέφει την απόσταση των σπιτιών **A** και **B**, αν ο Γιάννης θέλει να την αποκαλύψει, διαφορετικά −1.
4. Η συνάρτηση query(X) επιστρέφει την απάντηση που δίνει ο Γιάννης όταν ο Τάκης του υποδεικνύει το σπίτι **X**, δηλαδή το άθροισμα των αποστάσεων των **A**, **B** από το σπίτι **X**. Προφανώς, πρέπει να είναι $1 \leq \mathbf{X} \leq \mathbf{N}$.
5. Η συνάρτηση answer(A, B) πρέπει να καλείται στο τέλος κάθε στιγμιοτύπου του προβλήματος, όταν ο Τάκης έχει εντοπίσει τα κρυφά σπίτια **A**, **B**. Πρέπει να είναι $1 \leq \mathbf{A} < \mathbf{B} \leq \mathbf{N}$. Η συνάρτηση επιστρέφει true αν θα υπάρξει και επόμενο ανεξάρτητο

στιγμιότυπο του προβλήματος (με διαφορετικά **N**, **A**, **B**), ή false αν το πρόγραμμά σας πρέπει να σταματήσει την εκτέλεσή του.

Για το συγκεκριμένο πρόβλημα δεν χρησιμοποιούνται αρχεία εισόδου και εξόδου. Η επικοινωνία του προγράμματος σας με τον grader γίνεται μόνο μέσω των παραπάνω συναρτήσεων.

Παράδειγμα αλληλεπίδρασης:

Το πρόγραμμά σας καλεί	**Η συνάρτηση επιστρέφει**	**Περιγραφή**
`getN()`	`5`	Υπάρχουν 5 σπίτια, αριθμημένα από 1 έως 5.
`getD()`	`2`	Ας υποθέσουμε ότι τα κρυφά σπίτια βρίσκονται στις θέσεις 2 και 4 (φυσικά αυτό δεν το γνωρίζει ακόμα το πρόγραμμά σας). Η απόσταση των κρυφών σπιτιών είναι 2 και ο Γιάννης αποφασίζει να την αποκαλύψει.
`query(2)`	`2`	Η συνολική απόσταση των δύο σπιτιών από τη θέση 2 είναι 0+2=2.
`query(3)`	`2`	Η συνολική απόσταση των δύο σπιτιών από τη θέση 3 είναι 1+1=2.
`answer(2,4)`	`true`	Ο Τάκης μαντεύει σωστά πού βρίσκονται τα δύο σπίτια. Θα ακολουθήσει κι άλλο στιγμιότυπο.
`getN()`	`8`	Στο νέο στιγμιότυπο, υπάρχουν 8 σπίτια.
`getD()`	`3`	Ας υποθέσουμε ότι τα κρυφά σπίτια βρίσκονται αυτή τη φορά στις θέσεις 1 και 4. Η απόσταση των κρυφών σπιτιών είναι 3 και ο Γιάννης πάλι την αποκαλύπτει.
`query(8)`	`11`	Η συνολική απόσταση των δύο σπιτιών από τη θέση 8 είναι 7+4=11.
`query(5)`	`5`	Η συνολική απόσταση των δύο σπιτιών από την θέση 5 είναι 4+1=5.
`answer(1,4)`	`false`	Ο Τάκης πάλι μαντεύει σωστά πού βρίσκονται τα δύο σπίτια. Δεν θα ακολουθήσει άλλο στιγμιότυπο.

Περιορισμοί:

- $3 \leq N \leq 1.000$

- Το άθροισμα των N σε όλα τα στιγμιότυπα μίας περίπτωσης ελέγχου δεν θα ξεπερνά το 100.000.
- $1 \leq A < B \leq N$
- Αν παραβιάσετε το πρωτόκολλο επικοινωνίας, το πρόγραμμά σας θα τερματιστεί και θα θεωρηθεί ότι απέτυχε η περίπτωση ελέγχου. Ενδεικτικά και χάριν παραδείγματος, τα παρακάτω θεωρούνται παραβιάσεις του πρωτοκόλλου επικοινωνίας:
  - Κλήση της query ή της answer με παραμέτρους εκτός των ορίων που αναφέρονται.
  - Κλήση της query περισσότερες από Q φορές σε κάποιο στιγμιότυπο.
  - Κλήση της answer με λάθος τιμές των A και B.
  - Κλήση οποιασδήποτε συνάρτησης αφότου η answer επιστρέψει false.
  - Χρήση οποιασδήποτε εντολής εισόδου/εξόδου.

Υποπροβλήματα:

Στα τρία πρώτα υποπροβλήματα, ο Γιάννης θα αποκαλύπτει την απόσταση D των σπιτιών A και B.

1. (4 βαθμοί) D = 1, Q = 1.000
2. (6 βαθμοί) Q = 1.000
3. (20 βαθμοί) Q = 1

Στα υπόλοιπα υποπροβλήματα, ο Γιάννης δε θα αποκαλύπτει την απόστασή τους.

4. (6 βαθμοί) Q = 1.000
5. (9 βαθμοί) Q = 37
6. (12 βαθμοί) Q = 21
7. (17 βαθμοί) Q = 11
8. (26 βαθμοί) Q = 2

Παρατηρήσεις:

Μέγιστος χρόνος εκτέλεσης: 1 sec.
Μέγιστη διαθέσιμη μνήμη: 16 MB.

ΠΡΟΣΛΗΨΗ ΠΙΤΣΑΔΟΡΩΝ hiring

Ο Τάκης το πήρε τελικά απόφαση να προσλάβει εργαζόμενους. Μάλιστα, έχει βρει **N** υποψήφιους πιτσαδόρους. Κάθε υποψήφιος μπορεί να προσληφθεί με το πολύ ένα είδος συμβολαίου: χάλκινο, ασημένιο ή χρυσό. Ο προϋπολογισμός του Τάκη του επιτρέπει να προσφέρει μέχρι **X** χάλκινα, μέχρι **Y** ασημένια και μέχρι **Z** χρυσά συμβόλαια. Επιπλέον, ο ίδιος έχει φροντίσει να ισχύει ότι $\mathbf{X}+\mathbf{Y}+\mathbf{Z} \geq \mathbf{N}$.

Ανάλογα με το συμβόλαιο που λαμβάνει ένας πιτσαδόρος αποδίδει διαφορετικά. Αν στον i-οστο πιτσαδόρο προσφερθεί χάλκινο συμβόλαιο, αυτός θα έχει απόδοση $\mathbf{A_i}$, αν του προσφερθεί ασημένιο θα έχει απόδοση $\mathbf{B_i}$ και αν του προσφερθεί χρυσό θα έχει απόδοση $\mathbf{C_i}$. Για όλους τους υποψηφίους ισχύει ότι $\mathbf{A_i} \leq \mathbf{B_i} \leq \mathbf{C_i}$, δηλαδή όταν προσφέρουμε συμβόλαιο υψηλότερης βαθμίδας η απόδοση πάντα βελτιώνεται.

Προφανώς, ο Τάκης θέλει να μεγιστοποιήσει την συνολική απόδοση των πιτσαδόρων του, δηλαδή το άθροισμα των αποδόσεων τους. Παρόλα αυτά, είναι απαραίτητο να παραμείνει εντός προϋπολογισμού. Να βρείτε την μέγιστη συνολική απόδοση που μπορεί να πετύχει ο Τάκης, τηρώντας τα όρια που έχει θέσει για κάθε είδος συμβολαίου.

Πρόβλημα:

Να αναπτύξετε ένα πρόγραμμα σε μια από τις γλώσσες του ΠΔΠ (PASCAL, C, C++, Java) το οποίο θα διαβάζει τα δεδομένα εισόδου από το αρχείο **hiring.in** και θα εκτυπώνει τα αποτελέσματα στο αρχείο **hiring.out**.

Δεδομένα εισόδου (hiring.in):

Η πρώτη γραμμή περιέχει τον αριθμό του υποπροβλήματος (βλ. παρακάτω). Η δεύτερη γραμμή περιέχει τέσσερις ακέραιους αριθμούς **N**, **X**, **Y** και **Z**, χωρισμένους ανά δύο με ένα κενό διάστημα: το πλήθος των υποψηφίων, το μέγιστο πλήθος χάλκινων συμβολαίων, το μέγιστο πλήθος ασημένιων συμβολαίων και το μέγιστο πλήθος χρυσών συμβολαίων. Ακολουθούν **N** γραμμές καθεμία από τις οποίες αποτελείται από τρεις ακέραιους αριθμούς $\mathbf{A_i}$, $\mathbf{B_i}$ και $\mathbf{C_i}$, χωρισμένους

ανά δύο με ένα κενό διάστημα: την απόδοση του i-οστου υποψηφίου αν του προσφερθεί χάλκινο, ασημένιο ή χρυσό συμβόλαιο, αντίστοιχα.

Δεδομένα εξόδου (hiring.out):

Πρέπει να περιέχει μία μόνο γραμμή με έναν ακέραιο αριθμό: την μέγιστη συνολική απόδοση που μπορεί να πετύχει ο Τάκης αν προσφέρει μέχρι **X** χάλκινα, μέχρι **Y** ασημένια και μέχρι **Z** χρυσά συμβόλαια.

Παραδείγματα εισόδου - εξόδου:

hiring.in	hiring.out
1 5 3 1 1 3 6 8 1 1 2 4 9 12 3 5 7 9 9 9	31

Εξήγηση: Η καλύτερη επιλογή πιτσαδόρων είναι να δώσουμε ασημένιο συμβόλαιο στον πρώτο (απόδοση 6), χάλκινο στον δεύτερο (απόδοση 1), χρυσό στον τρίτο (απόδοση 12), χάλκινο στον τέταρτο (απόδοση 3) και χάλκινο στον πέμπτο (απόδοση 9). Η συνολική απόδοση έτσι είναι 6 + 1 + 12 + 3 + 9 = 31.

Περιορισμοί:

- $1 \leq N \leq 100.000$
- $0 \leq X, Y, Z \leq N$
- $X + Y + Z \geq N$
- $1 \leq A_i \leq B_i \leq C_i \leq 1.000.000.000$

Υποπροβλήματα:

1. (8 βαθμοί) $N \leq 14$
2. (24 βαθμοί) $Z = 0$
3. (3 βαθμοί) $X = 0$
4. (12 βαθμοί) $A_1 = A_2 = \ldots = A_N$ (οι αποδόσεις των χάλκινων συμβολαίων είναι όλες ίσες), $B_1 \leq B_2 \leq \ldots \leq B_N$ (οι αποδόσεις των ασημένιων συμβολαίων βρίσκονται σε αύξουσα σειρά) και

$C_1 \geq C_2 \geq \ldots \geq C_N$ (οι αποδόσεις των χρυσών συμβολαίων βρίσκονται σε φθίνουσα σειρά)

5. (20 βαθμοί) $Z = 1$
6. (33 βαθμοί) Δεν υπάρχουν περαιτέρω περιορισμοί

Παρατηρήσεις:

Μορφοποίηση: Στην είσοδο αλλά και στην έξοδο, κάθε γραμμή τερματίζει με έναν χαρακτήρα `newline`.

Μέγιστος χρόνος εκτέλεσης: 1 sec.
Μέγιστη διαθέσιμη μνήμη: 64 MB.