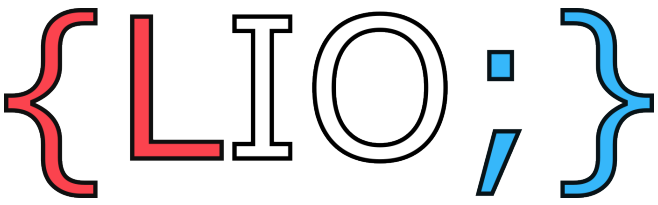
Lëtzebuerger Informatiksolympiad 2024

Qualifikationsrunde

Aufgabenstellung

Hinweise

- Die erlaubten Programmiersprachen sind Python 3, Java und C/C++.
- Alle Programme müssen in Form einer Konsolenanwendung abgegeben werden. Eine Anleitung, wie man eine Konsolenanwendung in den erlaubten Programmiersprachen erstellt, finden Sie in den Hinweisen auf der Seite `www.infosolympiad.lu` unter der Rubrik *The tasks*.
- Unter der Eingabe des Programms versteht man entweder die direkte Eingabe von Daten über die Tastatur oder die Weiterleitung aus einer Textdatei im Konsolenmodus. Unter der Ausgabe des Programms versteht man entweder die direkte Anzeige von Daten auf dem Bildschirm oder die Umleitung in eine Textdatei im Konsolenmodus.
- Die in den Ausführungsbeispielen gezeigten Formate der Ein- und Ausgabedaten müssen unbedingt beachtet werden.
- Zum Testen, Abgeben und Bewerten eines Programms muss die Quelldatei mit der Dateiendung `py`, `java` oder `c/cpp` in das automatisierte Online Portal CMS (Contest Management System) hochgeladen werden, das über die Homepage `www.infosolympiad.lu` oder direkt über die URL `http://158.64.46.20:81` erreichbar ist. Bitte verwende dein persönliches Login (Benutzername und Passwort), um auf dein Konto im CMS zuzugreifen. Der Dateiname der einzelnen Quelldatei muss mit dem Aufgabennamen identisch sein. Technische Einzelheiten zum Testen und Einreichen eines Programms findest du im CMS.
- Technische Details wie Zeit- und Speicherbegrenzungen sowie Kompilierbefehle entnehmen Sie bitte dem CMS.
- Du darfst über das CMS Fragen zu stellen. Jedoch werden die Antworten dir nicht beibringen, wie man eine Programmiersprache benutzt, noch dir erklären, wie du die Aufgaben unter Verwendung eines bestimmten Algorithmus lösen können. Die Fragen sollten im Zusammenhang mit dem CMS stehen oder Klärungsfragen zu den Aufgabenstellungen sein.



Vier gewinnt

Beschreibung

Diese Aufgabe basiert auf dem beliebten Zweispielerspiel "Vier gewinnt", auch bekannt als *Connect 4*, *Four Up*, *Plot Four*, *Find Four* usw. Weitere Informationen über dieses Spiel findest du auf Wikipedia. In dieser Aufgabe werden wir nicht das Spiel so implementieren, wie es gespielt wird (also dass die Spielsteine von unten aufgestapelt werden), sondern wir gehen einfach davon aus, dass alle Spielsteine (sowohl rote wie auch gelbe) bereits in einem quadratischen Spielfeld der Größe $N \times N$ platziert sind und dass wir einfach herausfinden müssen, ob es mindestens 4 zusammenhängende Spielsteine einer Farbe in einer Reihe, einer Spalte oder einer Diagonale gibt. Die Anzahl der Spielsteine jeder Farbe ist bei dieser Aufgabe irrelevant. Die Ausgabe des Programms hängt von der Anordnung der Spielsteine ab. Wenn es mindestens 4 zusammenhängende Spielsteine einer Farbe gibt, ist die Ausgabe diese Farbe. Wenn es sowohl für Rot als auch für Gelb mindestens 4 zusammenhängende Steine gibt, oder wenn es überhaupt keine 4 zusammenhängenden Steine derselben Farbe gibt, lautet die Ausgabe `NOPE` (siehe Ausführungsbeispiele).

Aufgabe

Schreibe ein Programm, das die richtige Antwort `RED`, `YELLOW` oder `NOPE` ausgibt für eine gegebene Anordnung der Spielsteine auf dem Spielfeld.

Beschränkung

- $N \in \mathbb{N}, 4 \leq N \leq 20$

Eingabe und Ausgabe

Eingabe

Die erste Zeile enthält die Größe N des Spielfeldes.
Die nächsten N Zeilen enthalten die Farben der Spielsteine im Gitter an, Zeile für Zeile. Der Großbuchstabe `R` steht für einen roten Spielstein, der Buchstabe `Y` steht für einen gelben Spielstein.

Ausgabe

Der Text `RED`, `YELLOW` oder `NOPE`, abhängig von der Anordnung der Steine auf dem Spielfeld.

Ausführungsbeispiel

Eingabe 1

```
6
RYYYRY
RYRRYY
YRYRYY
RRYRRR
RYRRYY
YRRYYR
```

Ausgabe 1

```
RED
```

Eingabe 2

```
5
YRYRY
RYRYY
YYRYY
RYYYR
YRYRY
```

Ausgabe 2

```
NOPE
```

Eingabe 3

```
5
YRYRY
YYRYY
RRYRR
YYYYY
RRRRR
```

Ausgabe 3

```
NOPE
```

Eingabe 4

```
7
RRYYRYR
YRYRYRY
YRYYYRY
RYRYYYR
RYYRRYY
YRRYYRY
YRRRRYR
```

Ausgabe 4

```
NOPE
```

Die obrigen Beispiele haben folgened Erklärungen:

- In Beispiel 1 gibt es vier $\mathtt{R}$ in Spalte 4
- In Beispiel 2 gibt es keine vier zusammenhängenden Spielsteine irgendeiner Farbe
- In Beispiel 3 enthält die unterste Reihe fünf aufgereihte $\mathtt{R}$, die vorletzte Reihe enthält fünf aufgereihte $\mathtt{Y}$ und es gibt auch Diagonalen, die vier zusammenhängende $\mathtt{Y}$ enthalten
- In Beispiel 4 gibt es vier $\mathtt{R}$ in der letzten Zeile, aber auch fünf $\mathtt{Y}$, die in Zeile 2 und Spalte 3 beginnen und diagonal nach unten verlaufen

Zeit und Speicherlimit

Aufgabenname	connect4
Eingabedatei	Standarteingabe
Ausgabedatei	Standartausgabe
Zeitlimit	1 Sekunde
Speicherlimit	256 Megabyte

Bemerkung

Dein Programm muss nicht überprüfen ob die Eingabe dem Format entspricht. Du kannst davon ausgehen dass sie immer gültig ist.

Quadrat- und Kubikzahlen

Beschreibung

Leo hat kürzlich Quadrat- und Kubikzahlen kennengelernt. (Eine Zahl n ist eine Quadratzahl, wenn $n = a^2$ für eine ganze Zahl a, und n ist eine Kubikzahl, wenn $n = b^3$ für eine ganze Zahl b.)
Hier sind einige Quadratzahlen, die Leo gefunden hat: 9, 25 und 81. Und hier sind einige Kubikzahlen, die er gefunden hat: 8, 27 und 125.
Es dauert nicht lange bis Leo sich langweilt beim Quadrat- und Kubikzahl suchen. Er will jetzt alle Zahlen kleiner als N finden, die sowohl Quadratzahlen als auch Kubikzahlen sind.
Ein Beispiel einer solchen Zahl ist $729 = 27^2 = 9^3$.

Aufgabe

Schreibe ein Programm, das alle Zahlen kleiner als N ausgibt, die sowohl Quadratzahlen als auch Kubikzahlen sind.

Beschränkungen

- $1 \leq N \leq 10^8$

Eingabe und Ausgabe

Eingabe

Die erste Zeile enthält eine ganze Zahl N.

Ausgabe

Eine Zeile mit allen Zahlen kleiner als N, die sowohl Quadratzahlen als auch Kubikzahlen sind (in aufsteigender Reihenfolge), getrennt durch Leerzeichen.

Ausführungsbeispiel

Eingabe

```
70
```

Ausgabe

```
1 64
```

Punkteverteilung

Subtask	Punkte	Beschränkungen
1	10	$N < 1000$
2	15	Keine weitere Beschränkung

Zeit- und Speicherlimit

Aufgabenname	squarecubes
Eingabedatei	Standarteingabe
Ausgabedatei	Standartausgabe
Zeitlimit	0.1 Sekunde
Speicherlimit	256 Megabyte

Letzflix

Beschreibung

Leo hat kürzlich von der Streaming-Plattform Letzflix erfahren. Um Filme auf Letzflix anzusehen, muss man ein Abonnement für C Euro abschließen, das K Tage lang gültig ist. (Das heißt, dass man in den nächsten K Tagen nach dem Kauf eines Abonnements so viele Filme ansehen kann wie man möchte). Wenn du ein Abonnement am Tag i kaufst, ist es ab dem Tag des Kaufs bis einschließlich Tag $i + K$ gültig.
Leo ist sehr gut organisiert und hat bereits einen Zeitplan erstellt, in dem er festgelegt hat, wann er sich N Filme ansehen will. Er weiß, dass er sich den i-ten Film am Tag a_i ansehen möchte.
Da Leo jedoch nicht gerne Geld verschwendet, möchte er, dass du ihm hilfst, so wenig Geld wie möglich für Abonnements auszugeben und trotzdem alle Filme in seinem Zeitplan zu sehen.

Aufgabe

Schreibe ein Programm, das berechnet, wie viel Geld Leo mindestens für Letzflix-Abonnements ausgeben muss, damit er seinen Zeitplan einhalten kann.

Beschränkungen

- $1 \leq N \leq 10^5$
- $1 \leq C \leq 100$
- $0 \leq K \leq 10^9$
- $1 \leq a_i \leq 10^9$

Eingabe und Ausgabe

Eingabe

Die erste Zeile enthält drei durch ein Leerzeichen getrennte Zahlen: N (die Anzahl der Filme), C (die Kosten des Abonnements), K (die Dauer des Abonnements).
Die zweite Zeile enthält N ganze Zahlen a_i, die den Tag beschreiben, an dem Leo den i-ten Film sehen möchte.

Ausgabe

Gebe eine einzige ganze Zahl aus: den Mindestbetrag, den Leo für Letzflix-Abonnements ausgeben muss.

Ausführungsbeispiel

Eingabe

```
5 2 4
1 3 6 11 13
```

Ausgabe

```
6
```

Erklärung

Leo muss an den Tagen 1, 6 und 11 jeweils 2 Euro für Abonnements ausgeben. Somit muss er insgesamt 6 Euro ausgeben.

Punkteverteilung

Subtask	Punkte	Beschränkungen
1	5	$K = 10^9$
2	5	$K = 0$ (Das Abonnement ist nur an dem Tag gültig, an dem man es kauft)
2	15	Keine weiteren Beschränkungen

Zeit- und Speicherlimit

Aufgabenname	letzflix
Eingabedatei	Standarteingabe
Ausgabedatei	Standardausgabe
Zeitlimit	1 Sekunde
Speicherlimit	256 Megabyte

Die Goldbachsche Vermutung

Beschreibung

Eines der ältesten ungelösten Probleme der Mathematik, die Goldbachsche Vermutung, besagt, dass jede gerade natürliche Zahl N größer als 2 als Summe zweier Primzahlen geschrieben werden kann. Die Vermutung sagt jedoch nichts darüber aus, wie viele solcher Primzahlpaare es für ein gegebenes N gibt. Das weckt dein Interesse und du beschließt, alle diese Paare zu zählen.

Aufgabe

Berechne für eine gegeben gerade ganze Zahl $N > 2$ die Anzahl der Paare von Primzahlen, deren Summe N ergeben.

Beispiel

Die Zahl 22 ist eine gerade positive ganze Zahl, die als Summe $3 + 19$, $5 + 17$ und $11 + 11$ von Primzahlen geschrieben werden kann. Somit gibt es 3 verschiedene Primzahlpaare für 22.

Beschränkungen

- $2 < N \leq 10^7$, N gerade

Eingabe und Ausgabe

Eingabe

Die erste Zeile enthält eine ganze Zahl, N.

Ausgabe

Dein Programm soll eine Zahl auf einer einzige Zeile ausgeben, nämlich die Anzahl der Primzahlpaare der Zahl N.

Ausführungsbeispiel

Eingabe

```
22
```

Ausgabe

```
3
```

Punkteverteilung

Unteraufgabe	Punkte	Beschränkungen
1	10	$N < 10000$
1	10	$N < 100000$
2	5	Keine weiteren Beschränkungen

Zeit- und Speicherlimit

Aufgabe name	goldbach
Eingabedatei	Standarteingabe
Ausgabedatei	Standartausgabe
Zeitlimit	1 Sekunde
Speicherlimit	256 Megabyte