Sitzdesign

Aus unerklärlichen Gründen wurde Florian vom Münchner Verkehrsverbund (MVV) beauftragt, die Sitze der U-Bahn neu zu gestalten. Sie wünschen sich eines modernes Design, das aber trotzdem nicht zu weit vom „klassischen Farbklecks-Muster“, so wie man es auch in Österreich von den Öffis kennt, abweicht. Dazu hat er sich folgendes überlegt:

Er geht aus von einer Zeichenkette der Länge n, bestehend aus von den Farben Rot (R), Grün (G) und Blau (B). Diese sollen das „Grundgerüst“ für sein Design bilden. Wiederholt schreibt er unter je zwei aufeinanderfolgenden Farben eine neue Farbe, basierend auf folgender Regel: Falls die beiden Farben dieselbe sind, so ist auch die neue Farbe gleich. Anderenfalls ergibt sich die neue Farbe als die fehlende der drei (R, G, B). Das macht er solange, bis nur noch eine Farbe übrig bleibt. Dadurch entsteht ein wundervolles Muster, wie man z.B. in Abbildung 1 sieht. Florian erhofft sich, dass dieses Muster auch beim MVV gut ankommt, und München bald voll von seinen Dreiecken ist.

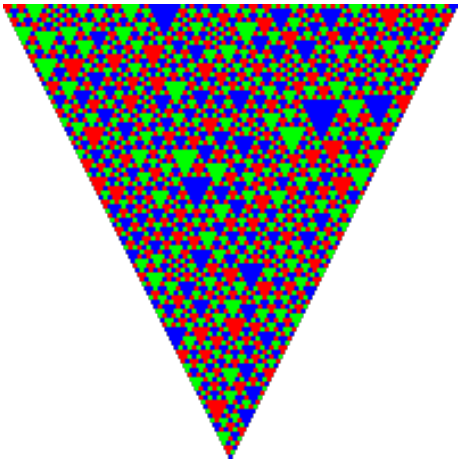

Abbildung 1: Florians Design

Während er mit verschiedenen Grundgerüsten herumprobiert, wundert sich Florian, wie die unterste Farbe von der Startsequenz abhängt. Kannst du ihm helfen für verschiedene Farbstrings berechnen, was die finale Farbe sein wird?

Implementierungsdetails

Implementiere die Funktion `solve(s)`, wobei `s` ein String - die Zeichenkette - mit Länge n ist. Gib in dieser Funktion die finale Farbe als ein einziges Zeichen zurück, entweder `R`, `G` oder `B`.

Beispiele

Eingabe	Ausgabe
GB	R

Eingabe	Ausgabe
RRR	R

Eingabe	Ausgabe
RGBG	B

Eingabe	Ausgabe
RBRGBRB	G

Subtasks

Allgemein gilt:

- $1 \leq n \leq 10^6$
- Der Farbstring besteht nur aus den Buchstaben R, G und B.

Subtask 1 (25 Punkte): $n \leq 1000$

Subtask 2 (35 Punkte): $n \leq 5 \cdot 10^4$

Subtask 3 (40 Punkte): Keine Einschränkungen

Limits

Zeitlimit: 0.5 s **Speicherlimit:** 256 MB

Eingabe

Das Einlesen der Eingabe wird von einem von uns geschriebenen Programm, dem Grader, übernommen. In der ZIP-Datei zu dieser Aufgabe findest du für jede Programmiersprache eine grader-Datei, das du zum Testen deiner Lösung mit deinem Programm kompilieren kannst.

Dieser Grader liest genau eine Zeile ein, die Farbsequenz, und gibt das Ergebnis der `solve`-Funktion auf der Standardausgabe aus.