Побудуйте два regex-и (що шукати й на що замінювати), які розв'язуватимуть такі задачі.

1. Побудувати пару reg. exp.-ів для пошуку фрагментів вигляду $\underbrace{\texttt{a}\ldots\texttt{a}}_{\geqslant 1}\underbrace{\texttt{b}\ldots\texttt{b}}_{\geqslant 2}\underbrace{\texttt{c}\ldots\texttt{c}}_{\geqslant 1}$ та заміни їх на `QQQ`.
2. Побудувати пару reg. exp.-ів, для знаходження усіх випадків, коли відразу після малої латинської літери йде велика латинська літера (без будь-якого іншого символу між ними) та виділення таких входжень знаками `_?_` з обох боків.

Наприклад, текст	має бути замінений на

`Jack lives in New York city.`
`Ulan-Ude is a city too, but Jack`
`doesn't want to live in Ulan-Ude.`
`"RegExp" is The Only STRing`
`in this file, where Reg.Exp.`
`should select a lower-case letter`
`followed with CAPITAL.`
	`Jack lives in New York city.`
	`Ulan-Ude is a city too, but Jack`
	`doesn't want to live in Ulan-Ude.`
	`"Re_?_gE_?_xp" is The Only STRing`
	`in this file, where Reg.Exp.`
	`should select a lower-case letter`
	`followed with CAPITAL.`

3. Побудувати пару reg. exp.-ів для заміни записів вигляду `$v_`⟨одна буква або цифра⟩`$` на `v[`⟨та сама буква або цифра⟩`]`. Наприклад, `$v_1$` має бути замінене на `v[1]`, `$v_s$` — на `v[s]`, і т. д.

Наприклад, текст	має бути замінений на

`Vertex $v_1$ is isolated ---`
`it's of degree~0.`
`Vertice $v_2$, $v_{7}$`
`and $v_{12}$ are terminal ---`
`each of them is of degree~1.`
 	
`BFS finds shortest paths`
`from a given vertex $v_{st}$`
`to all (reachable) vertice.`
 	
`In adjacency matrix, $A_{ij}$`
`can be either~1 (if there exists`
`an edge from $v_i$ to $v_j$)`
`or~0 (if no edge exists).`
 	
`$v_{12$ is not`
`a valid \TeX{} equation.`
	`Vertex v[1] is isolated ---`
	`it's of degree~0.`
	`Vertice v[2], $v_{7}$`
	`and $v_{12}$ are terminal ---`
	`each of them is of degree~1.`
	
	`BFS finds shortest paths`
	`from a given vertex $v_{st}$`
	`to all (reachable) vertice.`
	
	`In adjacency matrix, $A_{ij}$`
	`can be either~1 (if there exists`
	`an edge from v[i] to v[j])`
	`or~0 (if no edge exists).`
	
	`$v_{12$ is not`
	`a valid \TeX{} equation.`

«Одну букву або цифру» задавати як перелік `[\dA-Za-z]`, тобто або люба десяткова цифра, або люба буква від `A` до `Z` (велика) або від `a` до `z` (маленька). Більш коротким варіантом задання «букви або цифри» (як `\w`) в даному випадку користуватися не варто, бо `\w` задає також і символ "`_`", а зараз це непотрібно. Знак долара задавати як `\$`, тому що просто `$` означає «кінець рядка».

4. Аналогічно попередньому, але заміняти не односимвольні, а багатосимвольні індекси. При цьому, багатосимвольні індекси в початковому тексті додатково беруться у фігурні дужки, наприклад `$v_{12}$` (замінити на `v[12]`), `$v_{curr}$` (замінити на

v[curr]). Односимвольні індекси, взяті в дужки (наприклад, v_{7}) теж повинні замінятися, не взяті в дужки (як у попередьому завданні) — повинні не замінятися.

Наприклад, текст	має бути замінений на
Vertex v_1 is isolated --- it's of degree~0. Vertice v_2, v_{7} and v_{12} are terminal --- each of them is of degree~1. BFS finds shortest paths from a given vertex v_{st} to all (reachable) vertice. In adjacency matrix, A_{ij} can be either~1 (if there exists an edge from v_i to v_j) or~0 (if no edge exists). v_{12 is not a valid \TeX{} equation.	Vertex v_1 is isolated --- it's of degree~0. Vertice v_2, v[7] and v[12] are terminal --- each of them is of degree~1. BFS finds shortest paths from a given vertex v[st] to all (reachable) vertice. In adjacency matrix, A_{ij} can be either~1 (if there exists an edge from v_i to v_j) or~0 (if no edge exists). v_{12 is not a valid \TeX{} equation.

Фігурні дужки треба задавати як \{ та \}, тому що просто фігурні дужки мають інший смисл (діапазон кількості повторень).

5. Побудувати пару reg. exp.-ів, яка виконає заміни обох попередніх завдань (і одно-, і багатосимвольних індексів при v) за один прохід.

Наприклад, текст	має бути замінений на
Vertex v_1 is isolated --- it's of degree~0. Vertice v_2, v_{7} and v_{12} are terminal --- each of them is of degree~1. BFS finds shortest paths from a given vertex v_{st} to all (reachable) vertice. In adjacency matrix, A_{ij} can be either~1 (if there exists an edge from v_i to v_j) or~0 (if no edge exists).	Vertex v[1] is isolated --- it's of degree~0. Vertice v[2], v[7] and v[12] are terminal --- each of them is of degree~1. BFS finds shortest paths from a given vertex v[st] to all (reachable) vertice. In adjacency matrix, A_{ij} can be either~1 (if there exists an edge from v[i] to v[j]) or~0 (if no edge exists).

У неправильних рядках з порушенням відповідності фігурних дужок (наприклад, v_{12) можна хоч проводити заміну, хоч не проводити (і так і так вважатиметься правильним).

6. Побудувати пару reg. exp.-ів, яка за один прохід замінить усі фрагменти вигляду \texttt{...} (де ... — послідовність, що складається або з самих лише латинських літер, або з самих лише десяткових цифр) на фрагменти вигляду \begin{ttfamily}...\end{ttfamily} (де "..." у заміні — те, що було знайдене при

пошуку).

Наприклад, текст

`Variables \texttt{i},`
`\texttt{j} and \texttt{k}`
`are commonly used as integer counters.`
`Here, \texttt{R}, \texttt{R0}`
`and \texttt{R0after} are floating-point.`
`\texttt{a007} is a \texttt{string}.`
`\texttt{777} is an integer constant.`

має бути замінений на

`Variables \begin{ttfamily}i\end{ttfamily},`
`\begin{ttfamily}j\end{ttfamily} and \begin{ttfamily}k\end{ttfamily}`
`are commonly used as integer counters.`
`Here, \begin{ttfamily}R\end{ttfamily}, \texttt{R0}`
`and \texttt{R0after} are floating-point.`
`\texttt{a007} is a \begin{ttfamily}string\end{ttfamily}.`
`\begin{ttfamily}777\end{ttfamily} is an integer constant.`

7. Побудувати пару reg. exp.-ів, яка візьме в лапки всі послідовності з трьох (або більше) підряд слів, що починаються з великої літери. Звертати увагу слід тільки на перші літери, тобто несуттєво, великими чи малими будуть не-перші літери цих слів. Словами вважати послідовності, що складаються з самих лише латинських букв. Гарантовано, що в тексті не буде ніяких розділових знаків (ком, крапок, тире, тощо), крім пробілів між словами. Кількість пробілів між сусідніми словами може бути довільною, і це треба зберегти. Лапки треба ставити безпосередньо перед першим словом знайденого фрагменту та безпосередньо після останнього слова.

Наприклад, текст

`Jack lives in New York city`
`Ann lives in  New   York     CITY`
`A Sentence With Many Capital Case First Letters`
`A Sentence with Many Capital Case First Letters`
`A Sentence with Many Capital Case 1st Letters`
`CS106 Discrete Structures 2`
`106-th course of CS Discrete Structures II`

має бути замінений на

`Jack lives in New York city`
`Ann lives in  "New   York     CITY"`
`"A Sentence With Many Capital Case First Letters"`
`A Sentence with "Many Capital Case First Letters"`
`A Sentence with "Many Capital Case" 1st Letters`
`CS106 Discrete Structures 2`
`106-th course of "CS Discrete Structures II"`

8. Побудувати пару reg. exp.-ів для пошуку рядків вигляду `\circle{(`x, y`)` та заміни їх на `\circle{(`y, x`)` (тобто міняються місцями x та y). Гарантовано, що x та y — невід'ємні цілі числа; гарантовано також, що всередині даних рядків (які містять

\circle) нема жодного пробілу.

Наприклад, текст	має бути замінений на
\pen{2pt}	\pen{2pt}
\lines{(0,1000),(1010,1000)}	\lines{(0,1000),(1010,1000)}
\pen{0.5pt}	\pen{0.5pt}
\lines{(0,1010),(1010,1010)}	\lines{(0,1010),(1010,1010)}
\fillcolor{black}	\fillcolor{black}
\drawcolor{black}	\drawcolor{black}
\circle{(10,480),2}	\circle{(480,10),2}
\circle{(20,650),2}	\circle{(650,20),2}
\circle{(50,790),2}	\circle{(790,50),2}
\circle{(50,970),2}	\circle{(970,50),2}
\circle{(60,280),2}	\circle{(280,60),2}
\circle{(60,90)}	\circle{(90,60)}
\gfill\circle{(90,960),2}	\gfill\circle{(960,90),2}
\gfill\circle{(90,640),2}	\gfill\circle{(640,90),2}
\circle{(100,1010),2}	\circle{(1010,100),2}