Celem zadania jest napisanie prostego silnika przetwarzającego CSS-y. W ramach zadania należy wczytać ze standardowego wejścia sekcje CSS przeplatane sekcjami komend. Sekcje CSS należy sparsować i umieścić w odpowiednich strukturach, sekcje komend należy sparsować i wykonać wypisując na standardowe wyjście ew. rezultaty (po ==).

CSS

- Przetwarzanie rozpoczyna się wczytania deklaracji CSS. CSS jest syntaktycznie poprawny i składa się bloku atrybutów ew. poprzedzonych selektorami. Brak selektorów jest legalny (oznaczałby atrybuty aplikowane do wszystkiego).
- Selektory (selectors) są separowane przecinkami. Dopuszczalne są selektory legalne dla CSS, ale można założyć, że nie zawierają znaków przecinka ani nawiasów klamrowych.
- Blok atrybutów ujęty jest w nawiasy klamrowe.
- Atrybuty są oddzielone średnikami i składają się z nazwy (property) i wartości (value) oddzielonych dwukropkiem. Po ostatnim atrybucie w bloku może, ale nie musi nastąpić średnik.
- Jako wartości atrybutów mogą występować legalne dla CSS konstrukcje, jednak dla uproszczenia można bezpiecznie założyć, że ew. napisy nie są złośliwe tj. nie zawierają escapowanych znaków cudzysłowu, nawiasów klamrowych lub średników.
- Jeśli konkretny atrybut (nazwa) w bloku występuje więcej niż raz należy potraktować to jako jedno wystąpienie, przy czym znacząca jest ostatnia wartość.
- Zarówno selektory, nazwy atrybutów jak i wartości atrybutów nie wymagają interpretacji semantycznej tj. traktujemy je (po odrzuceniu skrajnych białych znaków), jako wartość. Tj. np. 'margin-left : 8px', 'margin: 4px 7px 4px 7px' traktujemy jako oddzielne, niezwiązane atrybuty o nazwach odpowiednio 'margin-left' i 'margin' i wartościach '8px' oraz '4px 7px 4px 7px' Podobnie, selektory są traktowane jako wartość i nie wymagają interpretacji tj. np.: 'h1' i 'h1.theme' traktujemy jako oddzielne, niepowiązane selektory.
- Uproszczenie: CSS nie zawiera komentarzy, ani selektorów typu @, bloki nie mogą się zagnieżdżać.
- Dla potrzeb większości testów (bez dużej straty można założyć, że żaden selektor ani atrybut nie jest podzielony na kilka linii (ciągle w jednej linii może być kilka separatorów/ i lub atrybutów).

Komendy.

W poniższych komendach i oraz j to dodatnie liczby całkowite (mieszczą się w int), natomiast n to legalna nazwa atrybutu.

???? - początek sekcji komend;

**** - wznów czytanie CSS;

? - wypisz liczbę bloków CSS;

i,S,? - wypisz liczbę selektorów dla bloku nr i (numery zaczynają się od 1), jeśli nie ma takiego bloku pomiń;

i,A,? - wypisz liczbę atrybutów dla bloku nr i, jeśli nie ma takiego bloku lub atrybutu pomiń;

i,S,j - wypisz j-ty selector dla i-tego bloku (numery bloków oraz atrybutów zaczynają się od 1) jeśli nie ma bloku lub selektora pomiń;

i,A,n - wypisz dla i-tej bloku wartość atrybutu o nazwie n, jeśli nie ma takiego pomiń;

n,A,? - wypisz łączną (dla wszystkich bloków) liczbę wystąpień atrybutu nazwie n. (W ramach pojedynczego bloku duplikaty powinny zostać usunięte na etapie wczytywania). Możliwe jest 0;

z,S,? - wypisz łączną (dla wszystkich bloków) liczbę wystąpień selektora z. Możliwe jest 0;

z,E,n - wypisz wartość atrybutu o nazwie n dla bloku z, w przypadku wielu wystąpień selektora z bierzemy ostatnie. W przypadku braku pomiń;

i,D,* - usuń cały blok nr i (tj. separatory+atrybuty), po poprawnym wykonaniu wypisz deleted;

i,D,n - usuń z i-tego bloku atrybut o nazwie n, jeśli w wyniku operacji pozostaje pusty blok powinien zostać również usunięty (wraz z ew. selektorami), po poprawnym wykonaniu wypisz deleted.

Uwagi implementacyjne:

Selektory oraz atrybuty powinny być przechowywane jako listy.

Poszczególne bloki CSS powinny być przechowywane w liście dwustronnej (aby efektywnie realizować komendę E – ostatnie wystąpienie atrybutu). Aby lepiej wykorzystać pamięć lista powinna obejmować tablicę T=8 struktur reprezentujących blok (gdzie T jest stałą możliwą do zmiany w czasie kompilacji) oraz licznik zajętych aktualnie struktur (z uwagi na ew. kasowanie elementów). Liczniki warto wykorzystać dla przyśpieszenia operacji parametryzowanych numerem komórki tj. i.

Przy alokowaniu nowego węzła tworzona jest tablica T-elementowa. Przy dodawaniu elementów, o ile jest wolne miejsce w węźle listy, należy je wykorzystać zanim alokowane zostaną nowe węzły. Jeżeli w przypadku usuwania elementów pozostanie pusta tablica należy węzeł usunąć. Nie trzeba przesuwać elementów miedzy węzłami, łączyć węzłów itd.

Przykład:

```
#breadcrumb
{
      width: 80%;
      font-size: 9pt;
}

h1, body {
      min-width: 780px;
      margin: 0;
      padding: 0;
      font-family: "Trebuchet MS", "Lucida Grande", Arial;
      font-size: 85%;
      color: #666666;
}

h1, h2, h3, h4, h5, h6 {color: #0066CB;}
????
?
1,S,?
1,S,1
1,A,?
2,A,font-family
h1,S,?
color,A,?
h1,E,color
1,A,padding
1,D,*
?
2,D,color
?

****
h1, h2, h3, h4, h5, h6 {color: #0066FF;}
????
?
```

Wynik:

```
? == 3
1,S,? == 1
1,S,1 == #breadcrumb
1,A,? == 2
2,A,font-family == "Trebuchet MS", "Lucida Grande", Arial
h1,S,? == 2
color,A,? == 2
h1,E,color == #0066CB
1,D,* == deleted
? == 2
2,D,color == deleted
? == 1
? == 2
```