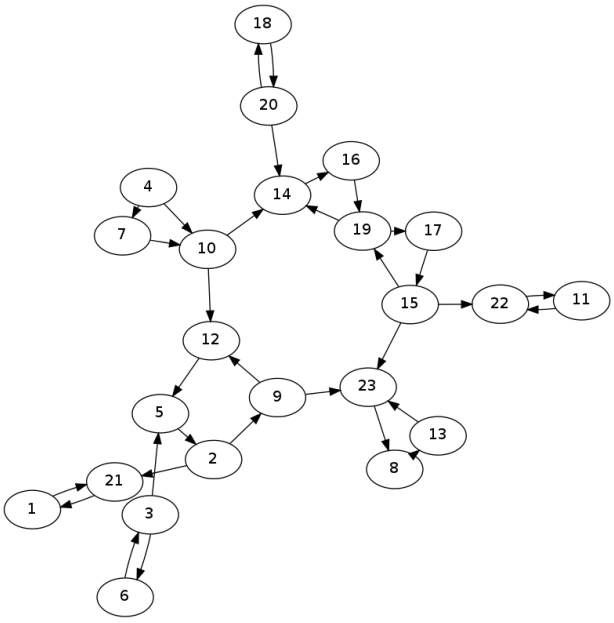
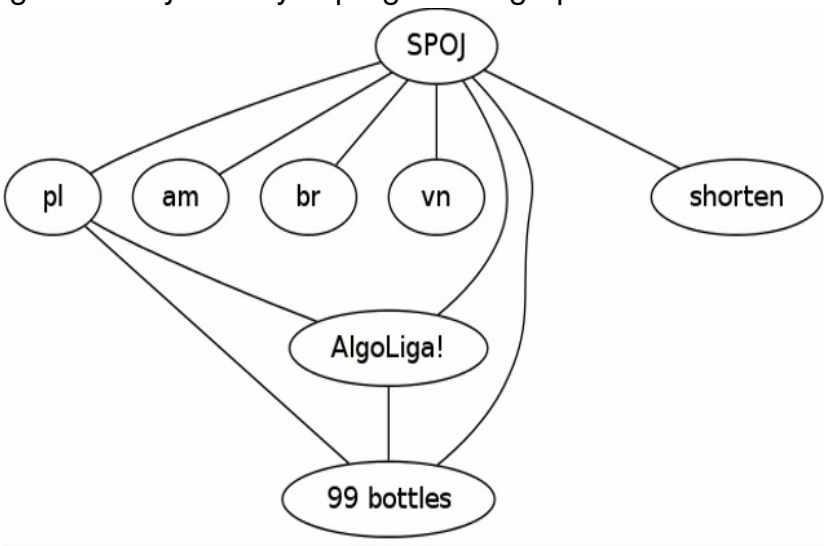
Rysowanie GRAFU



Jeżeli myślisz, że w tym zadaniu chodzi o narysowanie grafu, to nie pomyliłeś się. Dokładnie właśnie o to chodzi. Jeżeli myślisz, że jest to straszne zadanie i że jest to strasznie trudne, to być może masz rację. Ja też tak sądzę. Sądzę, że tak, to jest straszne zadanie i że tak, to jest strasznie, strasznie, strasznie...

Strasznie łatwe, pod warunkiem, że wiesz jak to zrobić. Ok. Na pewno zauważyłeś, że wśród różnych zadań na SPOJ'u są też zadania dotyczące GRAFÓW. Aby rozwiązać takie zadanie, warto najpierw narysować taki graf. Jeżeli ma on kilka węzłów i krawędzi, to żaden problem, można to zrobić nawet kredkami bambino. Gorzej, jeżeli chcemy narysować więcej węzłów [i krawędzi], a jednocześnie zależy nam na estetyce rysunku. To zadanie ma na celu pokazać jak to można [strasznie] łatwo zrobić. Potrzebny jest tylko darmowy program graphviz, minimalna wiedza jak go użyć oraz podstawowa znajomość języka dot http://en.wikipedia.org/wiki/DOT_language, potrzebnego do jego obsługi. Jeżeli nie chcesz instalować u siebie graphviz'a, zainteresuj się najnowszym Knoppix-Live DVD, bardzo dobrą dystrybucją Linuxa, gotową do użycia bez instalacji, która między innymi ma już zainstalowany ten pakiet i wiele innych przydatnych programiście narzędzi.

Powyżej i po prawej, przykłady prostych grafów narysowanych programem graphviz.



Poniżej przykład zapisu czterech grafów, w języku dot. W jednym pliku można umieszczać opisy wielu grafów. Podwójne "//" w języku dot oznaczają komentarz, podobnie jak w języku c/c++.

```
//1. Graf skierowany [directed graph]:

digraph {
a -> b;
b -> c;
c -> a;
}

//2. graf nieskierowany [undirected graph]

graph {
spoj1 -- spoj2;
spoj2 -- spoj3;
spoj3 -- spoj1;
}
```

```
//3. graf nieskierowany ważony  [undirected weighted graph]

graph {
1 -- 2 [label = 10];
2 -- spoj [label = -8];
spoj -- 1 [label = 8];
}

//4. graf skierowany ważony [directed weighted graph]

digraph {
1 -> 2 [label = 10];
2 -> 3 [label = -8];
3 -> 1 [label = 8];
}
```

Po zapisaniu powyższych 4 opisów grafów w pliku o dowolnej nazwie, np: first, uruchamiamy program neato [jeden ze składników pakietu graphviz]:

neato -Tpdf -O first [ENTER]

Natychmiast otrzymujemy rysunki zapisane w plikach:

first.pdf, first2.pdf, first3.pdf, first4.pdf

Jest jeszcze wiele, wiele innych możliwości i opcji języka dot [kształty węzłów, rodzaje linii, kolory, zapis obrazu w innych formatach], których nie wykorzystałem w tym zadaniu, a po ich opis odsyłam do manuala pakietu graphviz lub jego strony domowej programu. Czy zdobytą, przy rozwiązywaniu zadania, wiedzę i umiejętność wykorzystasz do rysowania grafów i zabawy z pakietem graphviz to już twoja sprawa. To zadanie pokazuje, że jest taka możliwość.

Twoim zadaniem jest napisanie programu konwertującego z formatu zapisu grafu spotykanego w zadaniach na SPOJU na format języka dot, zrozumiały dla programu graphviz. W zadaniu przyjąłem pewne ograniczenia, które nie obowiązują w języku dot.

Wejście

W pierwszej linii liczba grafów **t** (**1 ≤ t ≤ 10**). W nastepnej linii rodzaj grafu:

g - graf

d - graf skierowany

gw - graf ważony

dw - graf skierowany ważony

Następnie liczba krawędzi **n** (**1 ≤ n ≤ 20**). W kolejnych liniach opisy **n** krawędzi. Każdy opis to dwa indentyfikatory węzła początkowego i końcowego dla danej krawędzi. Dla grafów ważonych dodatkowo, jako trzeci parametr, waga krawędzi. Identyfikator węzła **id** nie przekracza **5** znaków (litery lub cyfry), a **waga** mieści się typie **int**.

Wyjście

Zapis grafu w formie czytelnej dla programu graphviz, zgodnie z opisem powyżej.

Przykład

Wejście:

```
2
d
3
1 2
2 3
3 1
gw
3
a b 9
b cccc 8
cccc a -7
```

Wyjście:

```
digraph {
1 -> 2;
2 -> 3;
3 -> 1;
}
graph {
a -- b [label = 9];
b -- cccc [label = 8];
cccc -- a [label = -7];
}
```

Dodatkowe informacje na temat pakietu GraphViz