Aufgabenblatt 4
Echtzeitsysteme (SoSe 2018)

Institut: Beuth Hochschule für Technik Berlin
Dozent: Prof. Dr. Christian Forler
Url: `https://lms.beuth-hochschule.de/`
Email: `cforler(at)beuth-hochschule.de`

Ressourcen-Belegungsgraph

Aufgabe 1 (8 Punkte) Deadlocks finden
Schreiben Sie ein Programm welches mittels der Tiefensuche einen Deadlock (Zyklus) in einem Ressourcen-Belegungsgraphen (RB-Graph) findet.

Der RB-Graph ist in einer Textdatei encoded.

- Die erste Zeile enthält die Anzahl Threads.
- Die zweite Zeile enthält die Anzahl Ressourcen.
- Alle folgenden Zeilen repräsentieren eine Kante welche aus drei Teilen besteht: 1) Art (`o`: Thread belegt (engl. *occupied*) Ressource; `w`: Thread wartet auf Ressource), 2) Startknoten und 3) Endknoten. Im Fall `o` handelt es sich bei dem Startknoten um eine Ressource und beim Endknoten um ein Thread. Im Fall `w` handelt es sich bei dem Startknoten um einen Thread und beim Endknoten um eine Ressource.

Erstes Beispiel: `$ cat graph1.txt`

```
3
3
o 1 1
o 2 2
w 1 2
w 2 3
w 3 1
o 3 3
```

Der Graph besteht aus 6 Knoten (drei davon sind Threads und drei Ressourcen) welche im Folgenden als $t1, t2, t3$ und $r1$, $r2$, $r3$ bezeichnet werden. Bei $t1, t2, t3$ handelt es sich um Threads und bei $r1$, $r2$, $r3$ um Ressourcen. Weiterhin besteht der Graph aus den folgenden 6 Kanten $(r1, t1), (r2, t2), (t1, r2), (t2, r3), (t3, r1)$ und $(r3, t3)$.

Hinweis: Implementieren Sie Ihr Graph als Adjazenzmatrix.
(siehe `https://de.wikipedia.org/wiki/Adjazenzmatrix`).

Zweites Beispiel: `$ cat graph2.txt`

```
7
6
o 1 1
o 1 3
o 1 5
o 2 2
o 4 4
o 6 6
o 5 7
w 1 2
w 3 2
w 4 2
w 4 3
w 2 3
w 6 2
w 5 5
w 7 4
```

Aufgabe 2 (0 Punkte) dot-Plot (Bonusaufgabe)

Ihr Programm aus Aufgabe 1 soll nun auch in der Lage sein den übergebenen RB-Graph als dot-Datei zu plotten. Informationen über das menschenlesbare, dot-Format finden Sie unter `http://graphs.grevian.org/reference`, `www.graphviz.org/pdf/dotguide.pdf` und `graphs.grevian.org/example`.

Mit dem Kommando `# dot -Tpdf myfile.dot -o myfile.pdf` können Sie aus dem dot-File ein PDF bzw. PNG rendern.

Beispielausgabe: `graph.dot`

```
digraph {
  r1 [shape=box, style=filled, fillcolor="orange"];
  r2 [shape=box, style=filled, fillcolor="orange"];
  r3 [shape=box, style=filled, fillcolor="orange"];
  t1 [style=filled, fillcolor="green"];
  t2 [style=filled, fillcolor="green"];
  t3 [style=filled, fillcolor="green"];
  r1 -> t1;
  r2 -> t2;
  t1 -> r2;
  t2 -> r3;
  t3 -> r1;
  r3 -> t3;
}
```