Programmierpraktiukum: **Winter 2023/24**
Algorithm Engineering **TU Berlin**
Schmid/Nichterlein/Figiel/Palarus **October 31, 2023**

2st Exercise Sheet

Vertex Cover I

Deadline: November 13
The task for the second submission is to implement an algorithm solving the Vertex Cover problem.

Vertex Cover
Input: An undirected graph $G = (V, E)$.
Output: Find a minimum size vertex cover, that is, a vertex subset $V' \subseteq V$ such that for each edge $e \in E$ at least one endpoint is in V' (i. e. $V' \cap e \neq \emptyset$).

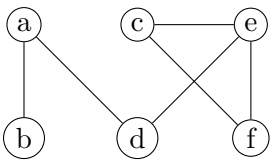
Input and output format

We use the following format for graphs (use stdin and stdout for input and output).

- Whitespaces at the beginning and end of every line are ignored. Text after # until the end of the line is ignored (treated as comment). Empty lines are ignored.
- The first non-comment line in the input contains two numbers separated by whitespace: n m. Here, n and m are the number of vertices and edges, respectively.
- All other lines define an edge by stating start and end vertex of the edge separated by a whitespace. Names of vertices may contain an arbitrary number of letters, numbers, and underscores.

Example:

Input graph:

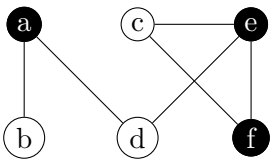


Input file:

```
6 6
a b
a d
c e
c f
d e
e f
```

The standard input for your program is a graph in the above described format. The output of the program is a minimum cardinality vertex cover. Every vertex should be written in one line (in an arbitrary order). For the above example this would be:

Vertex Cover (black vertices):



Output (only the solution):

```
a
e
f
```

Grading

The ranking will contain two submissions from us: A 50 % and a 100 % submission. Let I_{50} and I_{100} be the number of instances solved by these submissions, respectively ($I_{50} < I_{100}$). Your points are proportional to how many instances you are solving compared to I_{50} and I_{100}. If you solve I_{100} instances, you get the full 10 points; if you solve I_{50} instances, then you get 5 points. You cannot get more than 10 or less than 0 points.

For reference: Our 50 % submission is written in Java and implements the Degree-One Rule as well as the Maximum-Degree Branching using reasonable (but not overly optimized) data structures.