Aufgabe 4

a)

Zertifikat: Sei $V' = v_1, v_2, \cdots, v_k$. Dann konstruieren wir das Zertifikat im Form $c = bin(v_1)\#bin(v_2)\#\cdots\#bin(v_{k-1})\#bin(v_k)\#$. Die Länge des Zertifikat ist $O(n \log n)$ mit $n = |V|$, also poylnomiell in der Eingabelänge.

Verifizierer: 1. Prüfe zunächst, ob die Eingabe die richtige Form hat.
2. Prüfe dann, ob es für alle Paare von Knoten $(v, w) \in V' \times V'$ keine Kante $e = (v, w) \in E$ existiert:
- wir konstruieren eine Menge M' von allen Paaren von Knoten $(v, w) \in V' \times V'$: $M' = \{\{v_1, v_2\}, \{v_1, v_3\}, \cdots, \{v_{k-1}, v_k\}\}$. Schreibe M' auf Arbeitsband.
(Manche kann benachbart sein, manche nicht. Unsere Ziel ist, solche benachbarte Knoten zu finden)
- Iteriere aller paar von M' und suche, ob es einen Kanten in dieser Paare gibt ($(v, w) \in E$). Wenn ja, verwirft das Algorithmus. Wenn die Iterarion fertig ist, akzeptiert das Algorithmus.

Laufzeit:

- Fomatcheck: linear
- Konstruiere M'. Es gibt maximal $(\frac{(k-1)\cdot k}{2})$ mal Paare in M' mit $k = |V|$. So liegt die Laufzeit in $O(n^2)$.
  (z.B. in V' gibt es vier Knoten. Dann hat die Menge M' aus V' 6-mal Paare, also (1,2),(1,3),(1,4),(2,3),(2,4),(3,4), 6 = 3+2+1. Es kann in der Formel $(\frac{(k-1)\cdot k}{2})$ repräsentiert.)
- Bei Iteration kommt jede Paar (polynomiell viele, wie oben zeigt) von M' genau einmal vor.
  Für jede Paar muss noch mal über Kantenliste gelaufen werden (diese ist polynomiell lang).
- Insgesamt in Polynomialzeit.

Korrektheit: Wenn Verifizierer akzeptiert, dann lässt isch aus dem Zertifikat V' konsturieren und die Instanz ist k-Independent-Set.
Wenn die Instanz k-Independent-Set ist, lässt sich daraus ein gültiges Zertifikat konstruieren und der Verifizierer akzeptiert.

b)

```
1. K:={1,...,N}    // N=|V|
2. IF A(K) = FALSE THEN Ausgabe NotFound
3. FOR i:=1 to N do
        IF A(K\{i})=TRUE && |K|>1
        THEN K:=K\{i}
   ENDFOR
4. Ausgabe K
```

Die Laufzeit des Algorithmus ist in der Polynomialzeit beschränkt, weil FOR nut N-mal ausgefüht wird.

Korrektheit:
$G(= V, E) \in L \implies$ Der Algorithmus kann eine zulässige Lösung $K \in V$ ausgeben.
$\implies G$ ist k-Independent Set, $K \in V$
$\implies \forall (v, w) \in K \;:\; (v, w) \notin E$

Warum ist ausgegebenes K eine zulässige Lösung?

- Wir nehmen an, dass $\exists (v, w) \in K \;:\; (v, w) \in E$ gelte.

- D.h. es gibt mindesten einen Knoten, die mit anderen Knoten aus K benachbart ist.
  1. $\exists i \in K$, so dass $A(K\{i\}) = TRUE$.
     So ist es im Widerspruch dazu, dass der Algorithmus den Knoten i aus K gestrichen hat.
  2. $\forall i \in K$, so dann $A(K\{i\}) = FALSE$.
     Der Fall bedeutet, dass G kein k-Independent Set ist. So soll der Algorithmus kein K sondern ein NotFound am Anfang ausgeben. Somit liegt es auch im Widerspruch.
- Also gilt die Annahme nicht sondern ist das Aussage $\forall (v, w) \in K \ : \ (v, w) \notin E$ richtig.
- Somit ist ausgegebenes K eine zulässige Lösung.

Aufgabe 5

a)

Linker Graph: Ja. Knoten{1,2,5,7} mit Farbe 1 und Knoten{3,4,6} mit Farbe 2.
Rechter Graph: Nein. Denn falls Knoten 1 mit Farbe 1 gefärbt wird, dann muss Knoten 3 mit Farbe 2, Knoten 5 mit Farbe 1, Knoten 4 mit Farbe 2 und Knoten 7 mit Farbe 1, so dass $c(1) \neq c(7)$ nicht gilt.

b)

- Wir lösen das Problem mit einer Tiefensuche.
- In den Tiefensuche-Prozess färben wir alle Konten, indem jeder Knoten und seine Kinder verschiedene Farbe haben.
- Nachdem Tiefensuche-Prozess, suchen wir die Kanten, die nicht in der Tiefensuche sind. Vergleichen wir dann, ob jede Kante zwei verschiedenen gefärbten Knoten hat.
  Falls ja $\Rightarrow$ Der Graph gilt 2-COLORABILITY.
- Die Laufzeit von diesem Algorithmus ist polynomiell beschränkt (nach der Vorlesung).

c)

Nach der Vorlesung ist schon erkennt, dass COLORING $\leq_p$ SAT gilt. Deswegen hat COLORING einen Polynomialzeitalgorithmus.
Bei diesem Fall ist 3-COLORABILITY ein spezieller Fall von COLORING, so dass 3-COLORABILITY $\leq_p$ SAT. Weil SAT eine NP-Problem ist, ist 3-COLORABILITY auch ein NP-Problem.