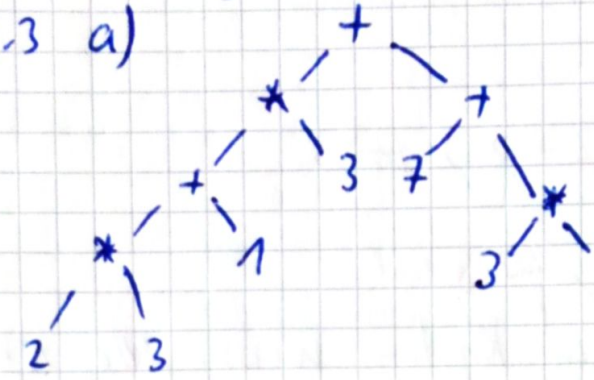
$6 + 5\frac{1}{2} + 8 + 7 + 7\frac{1}{2} = \frac{38}{}$

3.1 a) euf(amygnsich

b) ful amge sn cih

c) flugmaschine

3.3 a)



$$(((2*3)+1)*3)+(7+(3*3))$$
$$=((6+1)*3)+(7+(3*3))$$
$$=(7*3)+(7+(3*3))$$
$$=21+(7+(3*3))$$
$$=21+7+9$$
$$=37$$

b) Lösung mithilfe von Stack:

Idee: Das Eingabewort wird rückwärts durchlaufen, Zahlen werden in den Stack gelegt und Operatoren rechnen mit den Zahlen im Stack

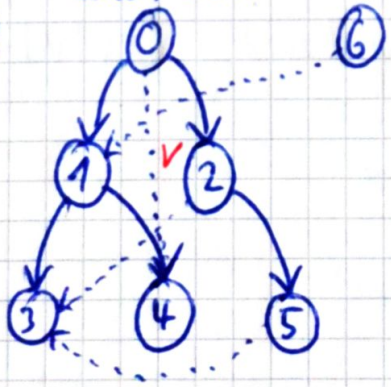
Pseudocode:

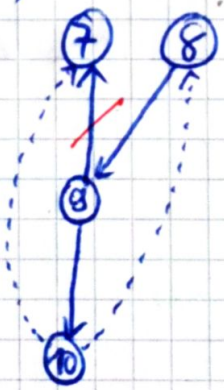
```
init stack S
for (int i=0; i<=|s|; i++) {
    if (s[|s|-i].isNumber) S.push(s[|s|-i])
    if (s[|s|-i].isOperator) S.operate(s[|s|-i])
}// wobei S.operate(*) z.B. die Multiplikation
 // der oberen beiden Elemente im Stack ausführt.
```

Laufzeit = $O(|A|)$

* S.print

3.4 a)





{(10,7),(10,8)} sind Rückwärtskanten

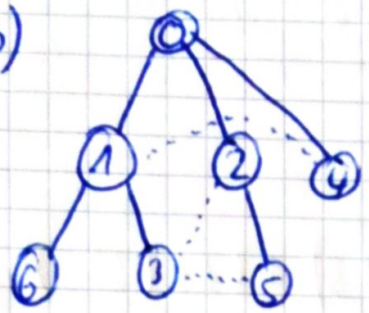
Legende?

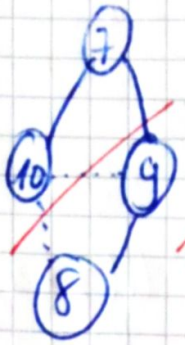
−1p.

{(0,1),(1,3),(1,4),(0,2),(2,5),(8,9),(9,7),9,10)}

Sind ~~nor male~~ Bäume Kanten. {(0,4),(2,3)} ~~(5,3)(6,1)~~ sind Querkanten.

b)





c) $0 \rightarrow 6 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 3$

3.5 Algorithmus: Nicht global $-\frac{1}{2}$ P.

Man färbt einen beliebigen Knoten weiss. Alle Knoten in der Adjazenzliste des nun weissen Knotens müssen schwarz gefärbt werden.
Alle Knoten in der Adjazenzliste der nun schwarz gefärbten Knoten müssen weiss gefärbt werden.
Man verfährt so solange, bis alle Knoten erfolgreich gefärbt wurden, dann ist der Graph zweifärbbar.
Sollte zwischendurch ein bereits gefärbter Knoten plötzlich andersfarbig gefärbt werden müssen, so liegt ein Konflikt vor, der Graph ist nicht zweifärbbar.
Da jeder Knoten mindestens ein Mal besucht wird und jede Kante mindestens ein mal entlanggegangen werden muss, ist die Laufzeit $\mathcal{O}(|V|+|E|)$

3.2 Pseudocode: Idee ? − 1p.

```
finde Strategie (baum) {
init kind = baum → LKind
falls baum → g == A dann True
falls baum → g == B dann False
falls baum → s == A dann {
    solange, wie (kind != 0) {
        falls (finde Strategie (kind))
            dann True
        kind = kind → RGeschwister
    } rand False...                    − ½p.
}
falls baum → s == B dann {
    solange, wie (kind != 0) {
        falls (finde Strategie (kind) == False)
            dann False
        kind = kind → RGeschwister
    }
}
```