Aufgabe 4

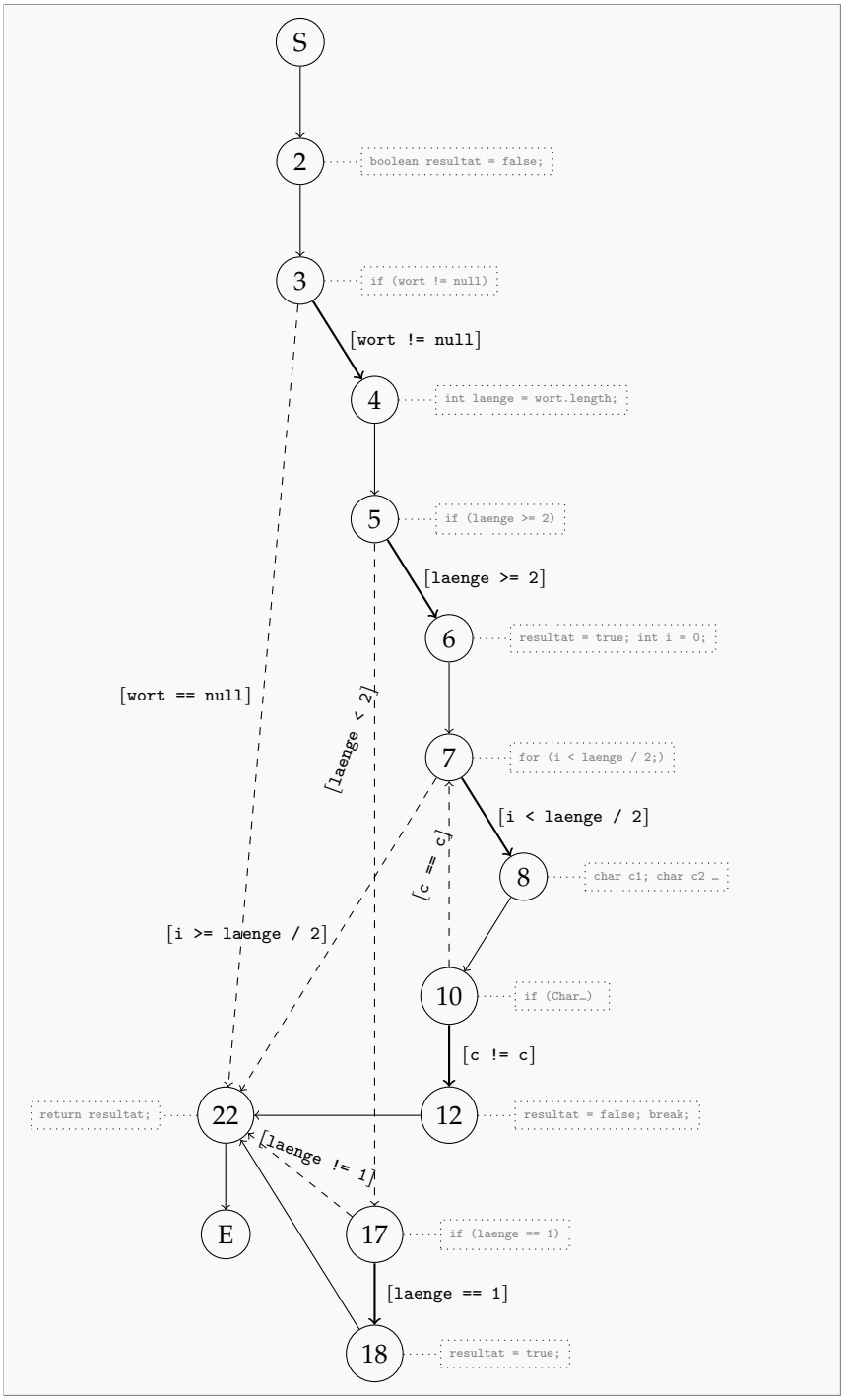
Diese Aufgabe behandelt *Wortpalindrome*, also Wörter, die vorwärts und rückwärts gelesen jeweils dieselbe Zeichenkette bilden, z. B. `Otto` oder `Rentner`. Leere Wortpalindrome (also Wortpalindrome der Wortlänge 0) sind dabei nicht zulässig.

Folgende *Java-Methode* prüft, ob das übergebene Zeichen-Array ein Wortpalindrom darstellt:

```
6   public static boolean istWortpalindrom(char[] wort) { // 1
7     boolean resultat = false; // 2
8     if (wort != null) { // 3
9       int laenge = wort.length; // 4
10      if (laenge >= 2) { // 5
11        resultat = true; // 6
12        for (int i = 0; i < laenge / 2; ++i) { // 7
13          char c1 = wort[i]; // 8
14          char c2 = wort[laenge - 1 - i]; // 9
15          if (Character.toLowerCase(c1) != Character.toLowerCase(c2)) // 10
16          { // 11
17            resultat = false; // 12
18            break; // 13
19          } // 14
20        } // 15
21      } else { // 16
22        if (laenge == 1) { // 17
23          resultat = true; // 18
24        } // 19
25      } // 20
26    } // 21
27    return resultat; // 22
28  } // 23
```

github: raw

(a) Geben Sie für die Methode einen *Kontrollflussgraphen* an, wobei Sie die Knoten mit den jeweiligen Zeilennummern im Quelltext beschriften.



(b) Geben Sie eine *minimale Testmenge* an, die das Kriterium der Anweisungsüberdeckung erfüllt.

Hinweis: Eine *Testmenge* ist *minimal*, wenn es keine Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität muss *nicht* bewiesen werden.

`isWortpalindrom(new char[] { 'a' }):`
Ⓢ - ② - ③ - ④ - ⑤ - ⑰ - ⑱ - ㉒ - Ⓔ

`isWortpalindrom(new char[] { 'a', 'b' }):`
Ⓢ - ② - ③ - ④ - ⑤ - ⑥ - ⑦ - ⑧ - ⑩ - ⑫ - ㉒ - Ⓔ

(c) Geben Sie eine *minimale Testmenge* an, die das Kriterium der *Boundary-Interior-Pfadüberdeckung* erfüllt.

Hinweis: Das Kriterium *Boundary-Interior-Pfadüberdeckung* beschreibt einen Spezialfall der Pfadüberdeckung, wobei nur Pfade berücksichtigt werden, bei denen jede Schleife nicht mehr als zweimal durchlaufen wird.

Es gibt noch ganz viele infeasible Pfade, die hier nicht aufgeführt werden.

Äußere Pfade

- `isWortpalindrom(null:`
  Ⓢ - ② - ③ - ㉒ - Ⓔ
- `isWortpalindrom(new char[] { }):`
  Ⓢ - ② - ③ - ④ - ⑤ - ⑰ - ㉒ - Ⓔ
- `isWortpalindrom(new char[] { 'a' }):`
  Ⓢ - ② - ③ - ④ - ⑤ - ⑰ - ⑱ - ㉒ - Ⓔ

Grenzpfade (boundary paths, boundary test) Für Schleifen fester Lauflänge ist diese Testfallgruppe leer.

`isWortpalindrom(new char[] { 'a', 'a' }):`
Ⓢ - ② - ③ - ④ - ⑤ - ⑥ - ⑦ - ⑧ - ⑩ - ⑦ - ㉒ - Ⓔ

`isWortpalindrom(new char[] { 'a', 'b' }):`
Ⓢ - ② - ③ - ④ - ⑤ - ⑥ - ⑦ - ⑧ - ⑩ - ⑫ - ㉒ - Ⓔ

Innere Pfade (interior test)

- `isWortpalindrom(new char[] { 'a', 'a', 'a', 'a' }):`
  Ⓢ - ② - ③ - ④ - ⑤ - ⑥ - ⑦ - ⑧ - ⑩ - ⑦ - ⑧ - ⑩ - ⑦ - ㉒ - Ⓔ
- `isWortpalindrom(new char[] { 'a', 'b', 'a', 'a' }):`
  Ⓢ - ② - ③ - ④ - ⑤ - ⑥ - ⑦ - ⑧ - ⑩ - ⑦ - ⑧ - ⑩ - ⑫ - ㉒ - Ⓔ

(d) Im Falle des Kriteriums Pfadüberdeckung können minimale Testmengen sehr groß werden, da die Anzahl der Pfade sehr schnell zunimmt. Wie viele *mögliche Pfade* ergeben sich maximal für eine Schleife, die drei einseitig bedingte Anweisungen hintereinander enthält und bis zu zweimal durchlaufen wird? Geben Sie Ihren Rechenweg an (das Ergebnis alleine gibt keine Punkte).

Pro Schleifendurchlauf: $2 \cdot 2 \cdot 2 = 2^3 = 8$

Maximal 2 Schleifendurchläufe: $2 \cdot 8 = 16$

(e) Könnte für das hier abgebildete Quelltext-Beispiel auch das Verfahren der *unbegrenzten Pfadüberdeckung* (also Abdeckung aller möglicher Pfade ohne Beschränkung) als Test-Kriterium gewählt werden? Begründen Sie.

Kante 7 nach 22