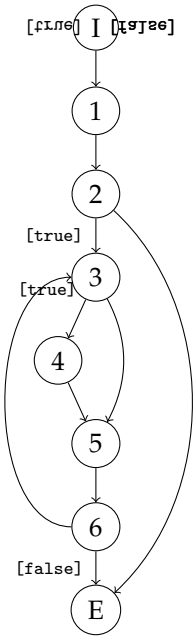
Aufgabe 2: „Testen“

Gegeben sei folgende Methode `isPalindrom` und ihr Kontrollflussgraph:
Abkürzungen: I = Import, E = Export

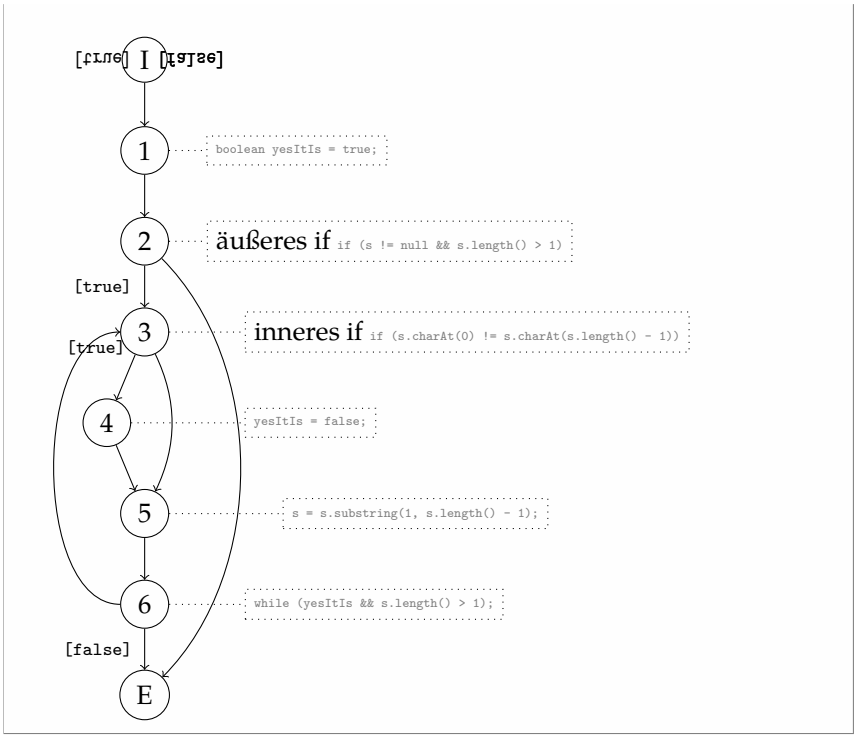
```
5   boolean isPalindrom(String s) {
6     boolean yesItIs = true;
7     if (s != null && s.length() > 1) {
8       do {
9         if (s.charAt(0) != s.charAt(s.length() - 1)) {
10          yesItIs = false;
11        }
12        s = s.substring(1, s.length() - 1);
13
14      } while (yesItIs && s.length() > 1);
15    }
16    return yesItIs;
17  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/sosy/pu_5/Aufgabe2.java`



(a) Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, die zum Erzielen einer vollständigen ... mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

> Bemerkung: In der Aufgabenstellung steht *„Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an,* [...] “. Das bedeutet, dass es hier erstmal egal ist, ob ein Pfad im Code möglich ist oder nicht!



(i) Verzweigungsüberdeckung (Branch-Coverage, C_1)

Pfad 1 (p1) ① - ① - ② - Ⓔ
(äußere `if`-Bedingung `false`)

Pfad 2 (p2) ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - Ⓔ
(äußere `if`-Bedingung `true`, innere `if`-Bedingung `false`, Wiederholung, innere `if`-Bedingung `true`, keine Wiederholung)

(ii) Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage, $C_{\infty,2}$)

ohne Ausführung der Wiederholung (äußere Pfade): p1 (siehe oben) ① - ① - ② - Ⓔ

Boundary-Test: (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)

interior-Test: (alle Pfade mit *einer Wiederholung des Schleifenrumpfes*; innerhalb des Schleifenrumpfes wieder alle Pfade!)
innere `if`-Bedingung `true`: ③ - ④ - ⑤ - ⑥
innere `if`-Bedingung `false`: ③ - ⑤ - ⑥

p5 ① - ① - ② - ③ - ④ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `true`, innere `if`-Bedingung `true`)

p2 (siehe oben) ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `false`, innere `if`-Bedingung `true`)

p6 ① - ① - ② - ③ - ④ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - Ⓔ

(innere `if`-Bedingung `true`, innere `if`-Bedingung `false`)

p7 ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `false`, innere `if`-Bedingung `false`)

mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

(b) Welche der vorangehend ermittelten Pfade für die $C_{\infty,2}$-Überdeckung sind mittels Testfällen tatsächlich überdeckbar (*„feasible“*)? Falls der Pfad ausführbar ist, geben Sie den zugehörigen Testfall an - andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

p1 `s = "a";`

p2 `s = "abaa";`

p3 `s = "ab";`

p4 `s = "aa";`

p5 nicht überdeckbar, da `yesItIs = false`, wenn innere `if`-Bedingung `true`) keine Wiederholung!

p6 nicht überdeckbar, da `yesItIs = false`, wenn innere `if`-Bedingung `true`) keine Wiederholung!

p7 `s = "abba";`

(c) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

$M = b + p = 3 + 1 = 4$

(b: Anzahl Binärverzweigungen, p: Anzahl Zusammenhangskomponenten)

Alternativ

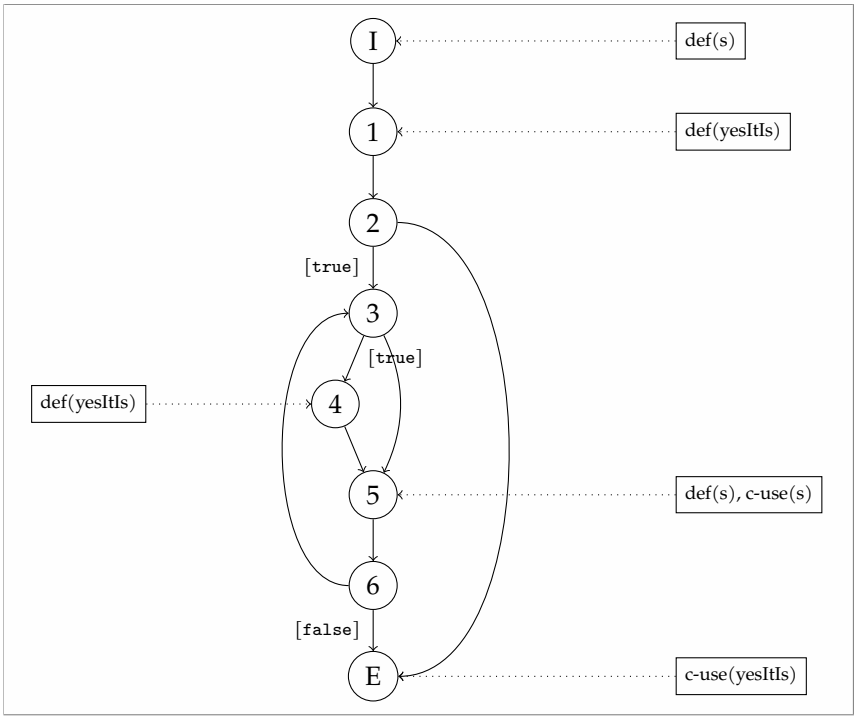
$M = e - n + 2p = 10 - 8 + 2 = 4$

(e: Anzahl Kanten, n: Anzahl Knoten, p: Anzahl Zusammenhangskomponenten)

(d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

Eine 100%-ige Pfadüberdeckung kann nicht erzielt werden, da es zum einen unüberdeckbare Pfade gibt (vgl. Teilaufgabe b). Zum anderen ist das Testen aller Testfälle nicht möglich, da die Anzahl an Zeichen des übergebenen Wortes nicht begrenzt ist und es somit eine unendliche Anzahl an Testfällen gibt.

(e) Übernehmen Sie den vorgegebenen Kontrollflussgraphen und annotieren Sie ihn mit allen relevanten Datenflussereignissen. Geben Sie jeweils an, ob die Verwendungen berechnend (`c-use`) oder prädikativ (`p-use`) sind.


I
def(s)
1
def(yesItIs)
2
[true]
3
[true]
def(yesItIs)
4
5
def(s), c-use(s)
6
[false]
E
c-use(yesItIs)