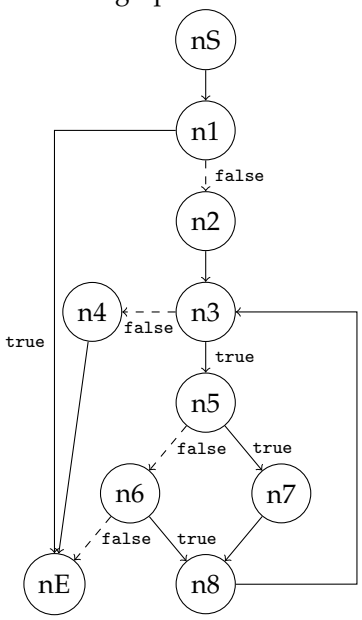
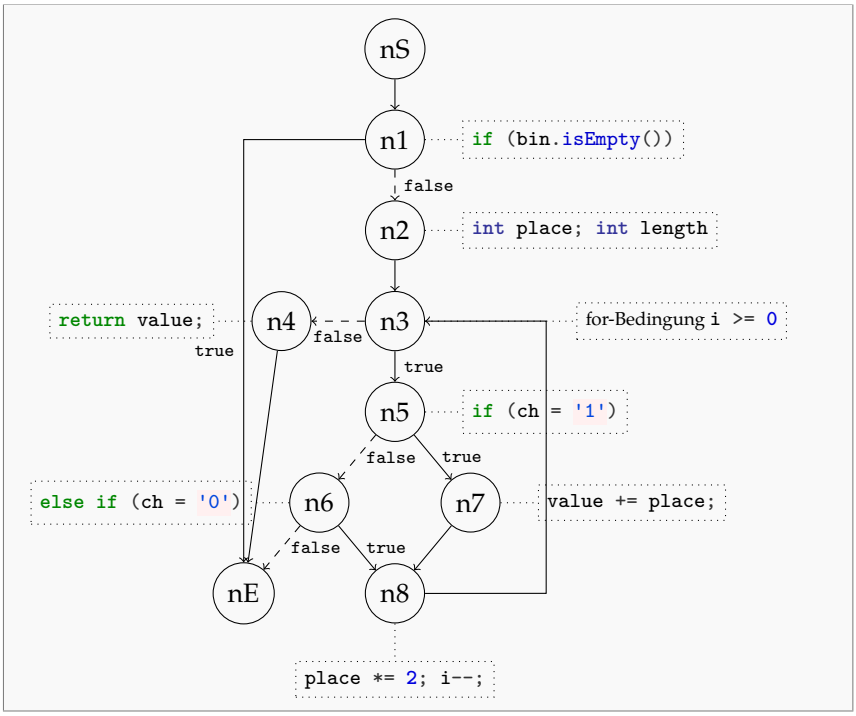
Aufgabe 5 (Check-Up)

Gegeben Sei folgende Methode und ihr Kontrollflussgraph:

```
4   int binToInt(String bin) {
5     if (bin.isEmpty())
6       return -1;
7     int place = 1, value = 0;
8     int length = bin.length() - 1;
9     for (int i = length; i >= 0; --i) {
10      char ch = bin.charAt(i);
11      if (ch == '1') {
12        value += place;
13      } else if (ch == '0') {
14        // do nothing
15      } else {
16        return -1;
17      }
18      place *= 2;
19    }
20    return value;
21  }
```



(a) Geben Sie je einen Repräsentanten aller Pfadklassen im Kontrollflussgraphen an, die zum Erzielen einer vollständigen



(i) Verzweigungsüberdeckung

p1 (Pfad 1) S 1 E
p2 S 1 2 3 4 E
p3 S 1 2 3 5 7 8 3 5 6 8 3 5 6 E

(ii) Schleife-Inneres-Überdeckung

Äußere Pfade (äußere Pfade): (ohne Ausführung der Wiederholung)

p1 S 1 E
p2 S 1 2 3 4 E

Grenzpfade (boundary test) (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)

p4 S 1 2 3 5 6 E

Innere Pfade (interior test) (alle Pfade mit *einer Wiederholung des Schleifenrumpfes*; innerhalb des Schleifenrumpfes wieder alle Pfade!)

p5 S 1 2 3 5 7 8 3 5 7 8 3 4 E
p6 S 1 2 3 5 7 8 3 5 6 8 3 4 E
p7 S 1 2 3 5 6 8 3 5 6 8 3 4 E
p8 S 1 2 3 5 6 8 3 5 7 8 3 4 E
p9 S 1 2 3 5 7 8 3 5 7 8 3 5 6 E
p10 = p3 S 1 2 3 5 7 8 3 5 6 8 3 5 6 E
p11 S 1 2 3 5 6 8 3 5 6 8 3 5 6 E
p12 S 1 2 3 5 6 8 3 5 7 8 3 5 6 E

mit minimaler Testfallanzahl genügen würden.

(b) Welche der vorangehend ermittelten Pfade sind mittels Testfälle tatsächlich überdeckbar („feasible“)? Falls der Pfad ausführbar ist, geben Sie bitte den Testfall an, andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

Erweitere Methode, die die Knotennamen ausgibt:

```
23  public static final String RESET = "\u001B[0m";
24  public static final String ROT = "\u001B[31m";
25  public static final String GRÜN = "\u001B[32m";
26
27  static int binToIntLog(String bin) {
28    System.out.println("\nInput: " + bin);
29    System.out.print("S");
30    System.out.print(1);
31    if (bin.isEmpty()) {
32      System.out.print("E");
33      System.out.println("\nOutput: " + -1);
34      return -1;
35    }
36    System.out.print(2);
37    int place = 1, value = 0;
```

```
38      int length = bin.length() - 1;
39      System.out.print(3);
40      for (int i = length; i >= 0; --i) {
41        char ch = bin.charAt(i);
42        System.out.print(5);
43        if (ch == '1') {
44          System.out.print(ROT + 7 + RESET);
45          value += place;
46        } else {
47          System.out.print(GRÜN + 6 + RESET);
48          if (ch == '0') {
49            // do nothing
50          } else {
51            System.out.print("E");
52            System.out.println("\nOutput: " + -1);
53            return -1;
54          }
55        }
56        System.out.print(8);
57        place *= 2;
58        System.out.print(3);
59      }
60      System.out.print(4);
61      System.out.print("E");
62      System.out.println("\nOutput: " + value);
63      return value;
64    }
65
66    public static void main(String[] args) {
67      binToIntLog(""); // p1
68      binToIntLog("??"); // p2 not feasible
69      binToIntLog("x01"); // p3
70      binToIntLog("x"); // p4
71
72      binToIntLog("11"); // p5
73      binToIntLog("01"); // p6
74      binToIntLog("00"); // p7
75      binToIntLog("10"); // p8
76
77      binToIntLog("x11"); // p9
78      binToIntLog("x01"); // p10
79      binToIntLog("x00"); // p11
80      binToIntLog("x10"); // p12
81    }
82  }
```

Alle mit Ausnahme von p2.

p2 ist nicht überdeckbar. Passiert ein Wert der Variable `bin` die erste if-Verzweigung, dann hat der Wert eine Länge größer 0 und betritt deshalb die Wiederholung mit fester Anzahl.

p1	S 1 E	`binToInt("");`
p2	S 1 2 3 4 E	not feasible
p3	S 1 2 3 5 7 8 3 5 6 8 3 5 6 E	`binToInt("x01");`
p4	S 1 2 3 5 6 E	`binToInt("x");`
p5	S 1 2 3 5 7 8 3 5 7 8 3 4 E	`binToInt("11");`
p6	S 1 2 3 5 7 8 3 5 6 8 3 4 E	`binToInt("01");`
p7	S 1 2 3 5 6 8 3 5 6 8 3 4 E	`binToInt("00");`
p8	S 1 2 3 5 6 8 3 5 7 8 3 4 E	`binToInt("10");`
p9	S 1 2 3 5 7 8 3 5 7 8 3 5 6 E	`binToInt("x11");`
p10 = p3	S 1 2 3 5 7 8 3 5 6 8 3 5 6 E	`binToInt("x01");`
p11	S 1 2 3 5 6 8 3 5 6 8 3 5 6 E	`binToInt("x00");`
p12	S 1 2 3 5 6 8 3 5 7 8 3 5 6 E	`binToInt("x10");`

(c) Bestimmen Sie anhand des Kontrollflussgraphen die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach Mc-Cabe.

Binärverzweigungen 4

Knoten 10

Kanten 13

Anhand der Binärverzweigungen:

$$\begin{aligned} M &= b + p \\ &= 4 + 1 \\ &= 5 \end{aligned}$$

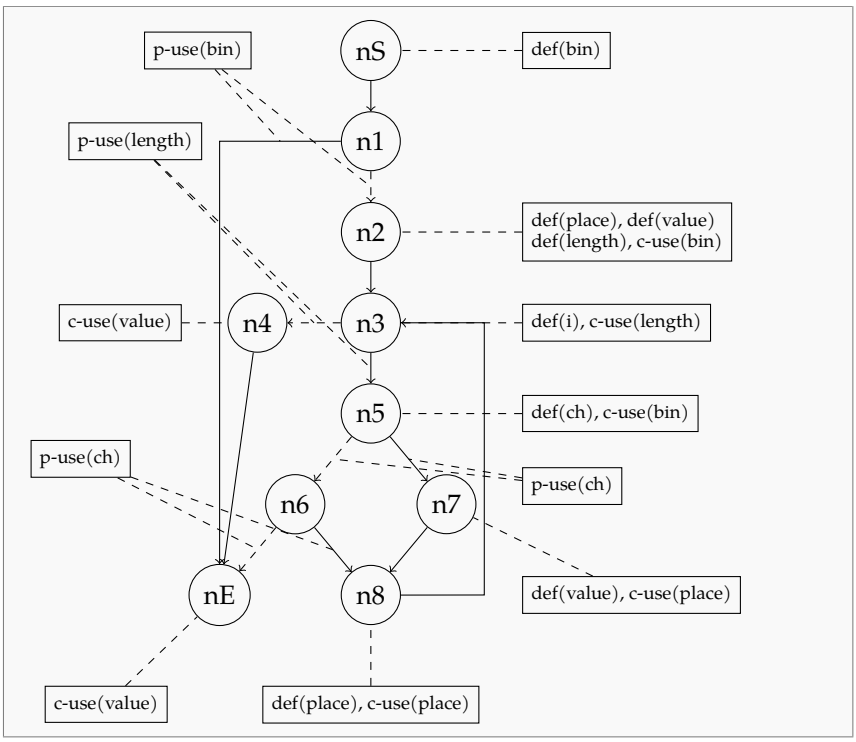
oder durch Anzahl Kanten e und Knoten n

$$\begin{aligned} M &= e - n + 2p \\ &= 13 - 10 + 2 \cdot 1 \\ &= 5 \end{aligned}$$

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

Nein, da **p2** nicht überdeckbar ist.

(e) Geben Sie zu jedem Knoten die jeweilige Datenfluss-Annotation (`defs` bzw. `uses`) für jede betroffene Variable in der zeitlichen Reihenfolge ihres Auftretens zur Laufzeit an.


p-use(bin)
nS
def(bin)
p-use(length)
n1
n2
def(place), def(value)
def(length), c-use(bin)
c-use(value)
n4
n3
def(i), c-use(length)
n5
def(ch), c-use(bin)
p-use(ch)
p-use(ch)
n6
n7
nE
n8
def(value), c-use(place)
c-use(value)
def(place), c-use(place)