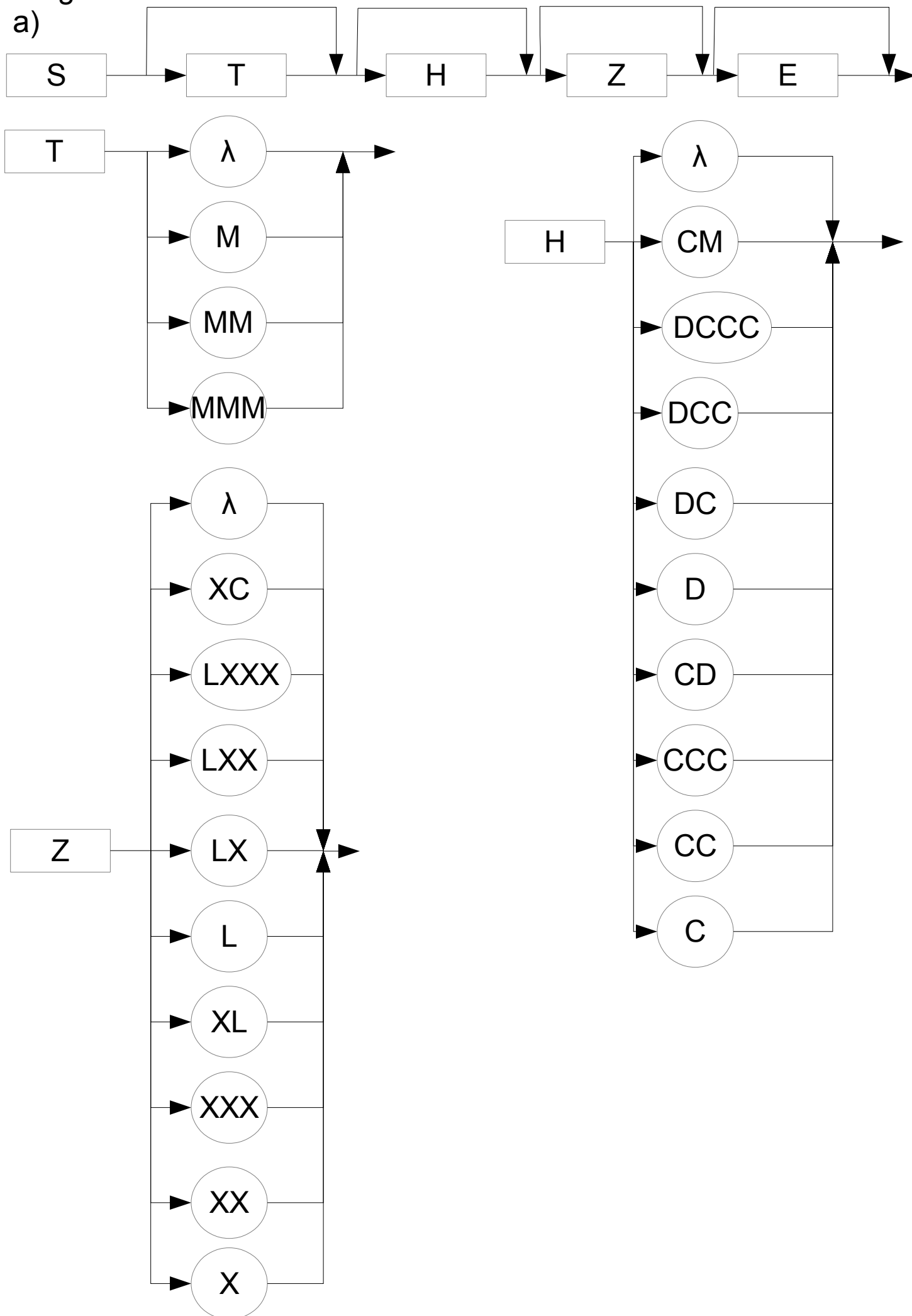
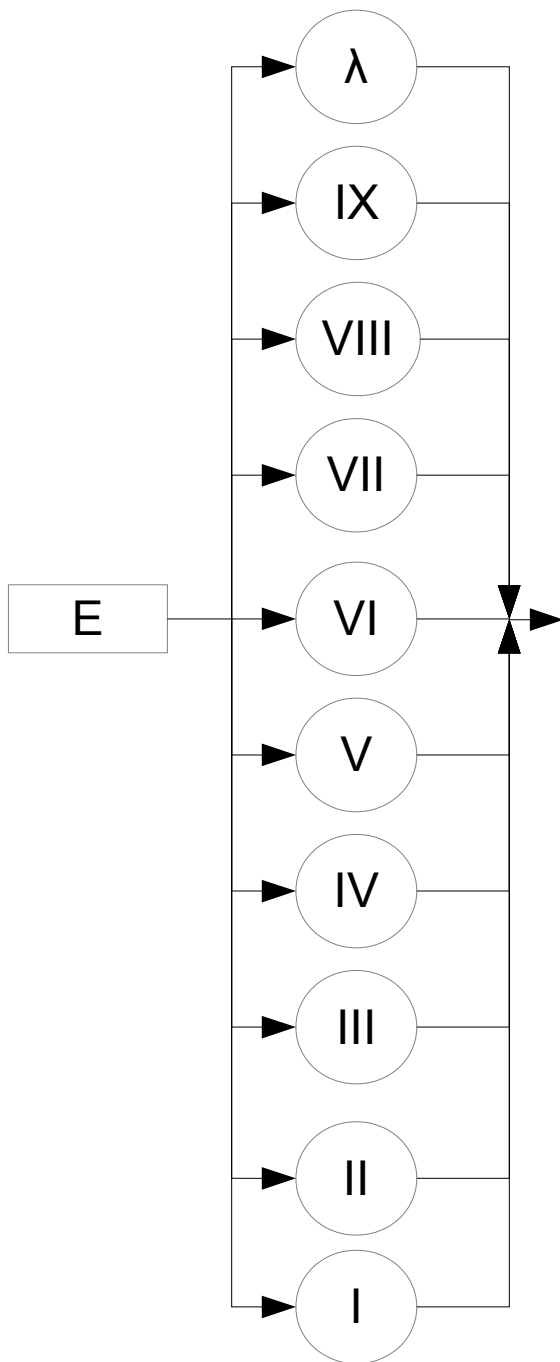


# Aufgabe 4.1

a)





**b)** In diesem Fall kann man sich einfach das Maximum der Anzahl der Terminal-Symbole jedes Nicht-Terminals nehmen, also **10**. Bei jedem Durchlauf kann man dann ein anderes dieser 10 Knoten durchlaufen, wobei man bei den anderen Nicht-Terminalen ebenfalls immer ein anderes nimmt.

**c)** Wie in a), nur dass jetzt noch die optionale Kante mitgenommen werden muss (die, die die Nicht-Terminals überspringen), da sonst für jedes Nicht-Terminal die Anzahl der Knoten und die Anzahl der Kanten gleich ist. Pro Nicht-Terminal hat man dann (Anzahl\_der\_Knoten + 1) Kanten und das Maximum ist **11**.

**d)** Alle Pfad-Möglichkeiten:  $6 \cdot 11 \cdot 11 \cdot 11 = \mathbf{7.986}$

```

01  /** Method to analyse the elements of a given list of strings. */
02  public boolean analyseList(List<String> in_lstList, int[] out_aiResults) {
03      boolean bResult = false;
04      if (in_lstList != null && !in_lstList.isEmpty()) {
05          // initialize counters
06          int iNumberCounter = 0;
07          int iEvenNumberCounter = 0;
08          int iWordCounter = 0;
09          int iEvenWordCounter = 0;
10          // go through all elements of the given list
11          for (String strElement : in_lstList) {
12              // ignore 'null' and empty strings
13              if (strElement != null && !strElement.isEmpty()) {
14                  int iNumber = 0;
15                  boolean bNumberFound = false;
16                  // try to parse the current list element to an int value
17                  try {
18                      iNumber = Integer.parseInt(strElement);
19                      bNumberFound = true;
20                  } catch (NumberFormatException e) {
21                      // do nothing
22                  }
23                  // check if a number was found
24                  if (bNumberFound) {
25                      iNumberCounter++;
26                      // check if number is even or odd
27                      if (iNumber % 2 == 0) {
28                          iEvenNumberCounter++;
29                          System.out.println(iEvenNumberCounter +
30                              ". even number found: " + iNumber);
31                      } else {
32                          System.out.println(iNumberCounter +
33                              ". number found: " + iNumber);
34                      }
35                  } else { // no number found
36                      iWordCounter++;
37                      // check if the number of characters of the current
38                      // list element is even or odd
39                      if (strElement.length() % 2 == 0) {
40                          iEvenWordCounter++;
41                          System.out.println(iEvenWordCounter + ". word " +
42                              strElement + " with even number of characters found: " +
43                              strElement);
44                      } else {
45                          System.out.println(iWordCounter + ". word found: " +
46                              strElement);
47                      } // end if
48                  } // end if
49              } // end if
50          } // end for
51          // put the results into output array
52          if (out_aiResults != null && out_aiResults.length >= 4) {
53              out_aiResults[0] = iNumberCounter;    // numbers found
54              out_aiResults[1] = iEvenNumberCounter; // even numbers found
55              out_aiResults[2] = iWordCounter;      // words found
56              out_aiResults[3] = iEvenWordCounter;  // words with even number
57              bResult = true;                        // ...of characters found
58          } // end if
59      }
60      return bResult;
61  }

```

