Aufgabe 1:
Schach hat endlich einen lang ersehnten Patch bekommen! Die Dame, die völlig op (over powered) war, wurde genervt (abgeschwächt). Früher konnte die Dame sich wie ein Turm und wie ein Läufer bewegen. Diese Tage sind endlich vorbei.

Nun kann die Dame sich wie ein **Turm**, ein **König** und ein **Springer** bewegen!

Daraus ergibt sich natürlich, dass das 8-Damenproblem erneut betrachtet werden muss. Wie viele Möglichkeiten gibt es 8 dieser neuen Damen auf einem 8×8 Schachbrett zu platzieren?

Nutzen Sie dafür den Code aus der Vorlesung 17, Seiten 17 und 19. Fügen Sie noch folgende main hinzu:

```
public static void main(String[] args) {
    LinkedList<int[]> res = new LinkedList<int[]>();
    setzeDameAll(res, 0);
    System.out.println(res.size());
}
```

Aufgabe 2:
Problem: **Longest Increasing Subsequence (LIS)**:

Gegeben sei eine Folge A von n natürlichen Zahlen $a_1, \ldots, a_n$. Gesucht ist die Länge der längsten Teilfolge von A, bei der die Folgenglieder streng wachsend sortiert sind.

Beachte den Unterschied zwischen Teilwort und Teilsequenz! Zur Bildung einer Teilsequenz schreitet man eine Sequenz von links nach rechts ab und kann für jedes Symbol entscheiden, ob es in die zu bildende Teilsequenz übernommen wird. Bei einem Teilwort wählt man lediglich den Start- und den End-Index.

Beispiel: $2, 4, 6, 1, 8, 3, 9$. Die längste streng aufsteigende Teilfolge ist offenbar $2, 4, 6, 8, 9$ und ist somit 5 Zeichen lang.

Implementieren Sie folgende Idee für ein dynamisches Programm zur Lösung dieses Problems und schätzen Sie dessen Laufzeitkomplexität ab.

- Jede Position ist für sich alleine ein Teilsequenz der Länge 1. Also ist die längste Teilsequenz bis zur Position $i = 0$ auch $L(0) = 1$.
- An einer Position i steht die Zahl a_i. Wenn wir diese an eine bisher berechnete Teilsequenz, die bis zur Position j maximal ist, anhängen wollen, ist es $1 + L(j)$. Das geht aber nur, wenn $j < i$ und $a_j < a_i$ gilt.
- Wir möchten a_i aber nicht an irgendeine bisher berechnete Teilsequenz anhängen. Wir möchten a_i an die bisher größte berechnete Teilsequenz anhängen. Somit müssen wir in $L(i)$ das Maximum aller möglichen Erweiterungen speicher.

$$L(i) = \begin{cases} 1 + max(L(j)|j < i \wedge a_j < a_i) & \text{wenn } 0 \leq i \leq n \\ 0 & \text{sonst} \end{cases}$$