Aufgabe 73:

a) Wenn wir insgesamt drei Sechsen werfen wollen, ist es äquivalent wie, wenn wir 3 Mal unabhängige Experimente durchführen und bei jedem eine 6 werfen.
E(X=1) = n * Pr[x = 6] = n * 1/6 <=> 1 = n * 1/6 → n = 6
d.h. E(X = 3) = 3 * 6 = 18
Im Erwartungswert muss man 18 Mal mit einem Würfel werfen, damit insgesamt drei Sechsen zu werfen

b) p = (n - 20 / n) ist die Wahrscheinlichkeit eine passende a zu wählen
E(X=1) = k * p <=> 1 = k * (n-20)/n → k = n/(n-20)
Erwartungswert für die Anzahl der Versuche ist n/(n-20)

Aufgabe 74:

Wir nehmen an, dass Suchen des k-kleinsten Schlüssels in O(n) schon effizient ist.

a) Skipliste:
Zuerst gehen wir auf die letzte Ebene (mit "Head-Element" anfangen) und bewegen wir uns Schritt für Schritt nach rechts, bis k-kleinsten Schlüssel gefunden wird. Da Skipliste immer sortiert ist, ist der k-kleinste Schlüssel auf der letzten Ebene doch das k-te Element.
→ Laufzeit = O(n) (Angenommen Zugriff auf die letzte Ebene ist O(lg n))

Außerdem könnten wir den Zeiger von der höchsten Ebene auf die 1. Ebene machen. Das spart uns in O(logn) Zeit und macht keinen Unterschied für andere Operationen.

b) Digitaler Suchbaum:
Initialisierung: Setzen wir counter gleich k ein.
Für jedes Blatt fügen wir eine Nummer hinzu, die zeigt, wie viele Kinder in dem Teilbaum unten gibt. Dann gehen wir von der Wurzel zuerst nach links und prüfen wie viele Elemente da unten gibt. Wenn es größer oder gleich counter ist, dann gehen wir wieder links und prüfen. Wenn es kleiner counter ist: counter = counter - Nummer (aktuelles Knoten) und dann gehen wir zu dem letzten solchen Knoten, der unbesuchte Kinder hat, zurück und da nach nächstem (von links nach rechts) Kind. Wiederholen solange, bis counter gleich 0 ist. Und dann ist der aktuelle Knoten (wenn es markiert ist) oder nächste (von links nach rechts) Knoten mit der Markierung genau der Schlüssel, den wir suchen.

Beim Einfügen eines Schlüssels gehen wir den ganzen Weg zur Wurzel zurück und inkrementieren Nummer jeden Knoten um 1. Das nimmt zusätzlich O(logn) Zeit.

Beim Löschen eines Schlüssels müssen wir Nummern aller Knoten, die auf dem Weg von diesem gelöschten Schlüssel zur Wurzel liegen um 1 dekrementieren, was auch zusätzlich O(logn) Zeit kostet.

c) Hashtabelle:

Wir nehmen an, dass die Hashtabelle mit offener Adressierung implementiert wird. Das heißt, in jeder Zelle gibt es höchstens nur ein Element.
Hashtabellen sind eher sich eignet für den schnellen Zugriff zum Schlüssel. Andere Operationen, wie Suchen von k-kleinstem Element, sind im Gegensatz relativ langsam.

Wir könnten zusätzlich die Nummerierung für jeden Schlüssel in der Tabelle hinzufügen. Nummer entspricht der Position des Schlüssels in sortierter Menge aller Schlüssel. Dann ist der k-kleinste Schlüssel der Schlüssel mit der Nummer k, Laufzeit: O(n).
Da wir beim Hinzufügen jedes Mal prüfen müssen, ob die Nummer stimmt und eventuell Nummerierung ändern (es nimmt zusätzlich O(log(n)) Zeit - vor jedem Einfügen Array schon sortiert (oder leer), dann einfach mit Binary Search suchen und hinzufügen). Für Löschen und Suchen macht es keinen Unterschied.(Für Löschen eines Elementes sollen wir alle die Nummer von anderen Elementen, deren Nummer größer als die Nummer von dem gelöschten Element ist, um 1 dekrementieren. Diese Implementation macht nur Sinn, wenn wir Suchen von k-ten Element öfter verwenden, als Hinzufügen von Element (Oder Elemente aus einer sortierten Liste hinzufügen)

d) 2-3-Baum:

Wir gehen den Baum in-Order-Weise durch und mitzählen, wie viele Schlüssel (Diese Erkennen wir als Elemente ohne Kinder oder einfach spezielle Markierungen) wir mithilfe einem counter durchgegangen sind, weil in einem 2-3 Baum Schlüsseln nur in Blättern gespeichert werden. Das k-te Element ist dann k-kleinster Schlüssel, da in-Order sortierter Folge entspricht.
→ Laufzeit: O(n)
Mit so einer Implementation ist schon diese Methode ziemlich effizient. Keine Änderungen der Struktur nötig.

Aufgabe 76:

a) Auf dem 8x8 Schachbrett kann man höchstens 6 Drachen, die sich nicht angreifen können, aufstellen. (Einstellungen: n = 8, dragons = 6)

b) Das kleinste Schachbrett ist 10 x 10 (mit 10 Drachen) (Einstellungen: n = 10, dragons = n)