Radixsort

Tema: Sorteringsalgoritmen radixsort.

U 1. Det finns en sorteringsmetod, positionssortering (eng. Radix sort) som sorterar en mängd icke-negativa heltal med ett visst känt maximalt antal siffror mycket snabbt genom att använda köer. (Egentligen är metoden inte begränsad till att sortera tal, den klarar även följder av annat slag t ex strängar.) Algoritmen använder sig (för sortering av heltal) av 10 köer, en för varje siffra 0..9, samt en kö som från början innehåller de osorterade talen och som i slutet av algoritmen innehåller talen sorterade i växande ordning. Den arbetar enligt följande, där vi använder `LinkedList`-klassen för att representera köerna:

```
Placera talen i en kö numberQ av typ LinkedList<Integer>
for (int i = 1; i <= d; i++) {  // d är antalet siffror i det längsta talet
    så länge numberQ inte är tom
        tag ut det första talet ur numberQ;
        j = i:e siffran bakifrån i talet;
        placera talet i kön subQ[j];
    for (int j = 0; j < 10; j++)    // konkatenera (den nu tomma) numberQ
        numberQ.append(subQ[j]);    // med de 10 köerna i subQ
}
```

I algoritmen distribuerar man först talen på de tio köerna med avseende på den sista siffran, sedan med avseende på den näst sista, osv. Efter varje sådan distribution konkateneras de tio köerna. Försök övertyga dig om att algoritmen är korrekt genom att studera exemplet som finns på sista sidan i denna laboration. Implementera metoden `radixSort`:

```
/** Sorterar talen i vektorn a med positionssortering. d anger maximalt
    antal siffror i talen. */
    public static void radixSort(int[] a, int d);
```

Tips: Siffror ur ett heltal kan extraheras med hjälp av divisionsoperatorn / och modulooperatorn %. När ett heltal a divideras med ett annat heltal b med operatorn /sker heltalsdivision vilket innebär att decimalerna stryks efter divisionen. Om a är ett heltal blir $a/10$ därför det tal som består av alla siffror i a utom den sista. Operatorn % står för rest vid heltalsdivision. Sista siffran i ett tal a får man därför ur $a\%10$. Näst sista siffran i a är den sista i talet $a/10$ dvs vi får den genom att bilda $a/10\%10$. Allmänt gäller att i-te siffran från slutet kan erhållas ur $a/10^{i-1}\%10$.

Gör så här för att skapa de tio köerna i subQ:

```
LinkedList<Integer>[] subQ = (LinkedList<Integer>[]) new LinkedList[10];
for (int i = 0; i < 10; i++) {
    subQ[i] = new LinkedList<Integer>();
}
```

Den första satsen skapar en vektor av typ `LinkedList<Integer>[]`. I `for`-satsen initieras de tio elementen i vektorn som tomma köer av typen `LinkedList<Integer>`. Man förväntar sig kanske att den första programsatsen skulle kunna skrivas enklare:

```
LinkedList<Integer>[] subQ = new LinkedList<Integer>[10]; //fel!
```

Det är emellertid inte tillåtet att i Java skapa vektorer vars element är av en parametriserad typ. För att kunna skapa vår vektor måste vi därför först skapa en vektor `LinkedList[]` (utan att ange typen av element) och sedan göra typkonvertering till den typ vi önskar.

Exempel på användning av RadixSort.

Nedan ges ett exempel på hur en följd av heltal sorteras med metoden RadixSort. Talen som sorteras är 721, 10, 51, 122, 674, 96, 109, 44, 236, 178, 1, 567, 674. Vi börjar med att distribuera talen med avseende på den sista siffran på 10 köer numrerade 0,1,..9 Resultatet av detta blir att köerna får följande innehåll:

```
0:      10
1:      721 51 1
2:      122
3:
4:      674 44 674
5:
6:      96 236
7:      567
8:      178
9:      109
```

Därefter konkateneras dessa köer (i ordningen 0, 1,.. 9) och vi får den resulterande kön: 10 721 51 1 122 67 44 674 96 236 567 178 109

Det vi nu åstadkommit är att talen är sorterade med avseende på sin sista siffra. Talen i denna kö distribueras nu med avseende på den andra siffran från slutet på 10 köer på samma sätt. Observera att ensiffriga tal då hamnar i kö nummer 0. Den andra siffran från slutet i dessa motsvarar alltså en inledande nolla i talet, vilket också ges av den formel för att räkna fram i:e siffran från slutet som anges i uppgiften:

```
0:      1 109
1:      10
2:      721 122
3:      236
4:      44
5:      51
6:      567
7:      674 674 178
8:
9:      96
```

Dessa köer konkateneras nu med resultatet: 1 109 10 721 122 236 44 51 567 674 674 178 96

Lägg märke till att om vi enbart betraktar de tal som består av de två sista siffrorna så utgör dessa nu en sorterad följd. Eftersom det maximala antalet siffror i talen är tre behövs ett sista tredje pass i vilket talen igen distribueras, nu med avseende på den tredje siffran från slutet. (Nu hamnar alla tal som har en eller två siffror i kö nummer 0):

```
0:      1 10 44 51 96
1:      109 122 178
2:      236
3:
4:
5:      567
6:      674 674
7:      721
8:
9:
```

Efter konkatenering får vi nu det sorterade slutresultatet: 1 10 44 51 96 109 122 178 236 567 674 674 721