This handout gives the **COUNTINGSORT** and **RADIXSORT** algorithms we covered in class, including how to implement **COUNTINGSORT** for arbitrary records with integer keys.

**COUNTINGSORT** sorts an array of $n$ integers all of which lie in the range $0 \ldots k - 1$. **RADIXSORT** sorts an array $n$ of $d$-digit numbers (where “digits” can be represented in any base $k$, e.g. 10, 2, 7, etc).

Remember that **RADIXSORT** works from *least* to *most* significant digit. As we write numbers, this ordering is right-to-left. In base 10, it means that we first sort on the “ones” position, then on the “tens” position, then on the “hundreds” position, etc.
**CountingSort** $(A, n, k)$

\[
\text{for } j \text{ in } 0 \ldots k - 1 \text{ do }
\]
\[
\text{counts}[j] \leftarrow 0
\]

\[
\text{for } i \text{ in } 0 \ldots n - 1 \text{ do }
\]
\[
\text{counts}[A[i]]++
\]

\[
i \leftarrow 0
\]

\[
\text{for } j \text{ in } 0 \ldots k - 1 \text{ do }
\]
\[
\text{for } m \text{ in } 1 \ldots \text{counts}[j] \text{ do }
\]
\[
A[i] \leftarrow j
\]
\[
i++
\]

**RadixSort** $(A, d, n, k)$

\[
\text{for } i \text{ in } 1 \ldots d \text{ do } \quad \triangleright \text{from least to most significant}
\]
\[
\text{sort records by } i\text{th digit}
\]
\[
\text{with CountingSort}(A, n, k)
\]
To implement `COUNTINGSORT` for arbitrary records with integer keys in the range $0 \ldots k - 1$, here is some pseudocode.

```
COUNTINGSORT(A, n, k)
    for $j$ in $0 \ldots k - 1$
        counts[$j$] ← 0

    for $i$ in $0 \ldots n - 1$
        counts[A[$i$].key]++

    $P[0] ← 0$
    for $i$ in $1 \ldots k - 1$
        $P[i] ← P[i - 1] + \text{counts}[i - 1]$

    for $j$ in $0 \ldots n - 1$
        $i ← A[j].key$
        $B[P[i]] ← A[j]$
        $P[i] += 1$
    copy $B$ into $A$
```