Permutering:

Dette er en forklaring av algoritmen for koden i **EKS_19_Permutering.cpp/java**:

For å hjelpe oss med å bytte rundt på *alle* elementene i arrayen `arr` på *alle tenkelige måter*, (dvs. permutere) så trenger vi to hjelpefunksjoner:

- `bytt(a, b)` – bytter innholdet av parameterne `a` og `b`
- `roterVenstre(arr, i, n)` – *venstreroterer ett* hakk innholdet i `arr` mellom indeksene `i` og `n`

(Venstrerotering = gjøre en forskyvning *ett* hakk mot venstre/ned av alle elementene i posisjon `i` til posisjon `n`, og flytter det som stod i posisjon `i` opp til posisjon `n`.)

Algoritmen:

Hovedideen er at når vi kaller den rekursive funksjonen `permuter(arr, i, n)` så har dette til oppgave å generere *alle* permutasjoner av de elementene som i kalløyeblikket er i `arr` mellom indeksene `i` og `n`.
Dette gjøres ved å flytte *alle etterfølgende* elementer *etter* nr. `i` ned i posisjon nr. `i` (vha. `bytt(...)`). *For hver gang* kaller den seg selv rekursivt slik at dette også gjøres for indeks `i+1`. Når denne parameteren har blitt `n`, er det ikke flere elementer igjen å permutere, og vi kan bruke/skrive ut `arr`.
I tillegg *skal* funksjonen sørge for at når den avsluttes, så *må* elementene stå i *nøyaktig samme rekkefølge* som da den ble kalt. Dette gjøres vha. `venstreRoter(arr, i, n)`.

Eksempel:

Om `n` = 6, så gjøres følgende når `permuter(arr, 2, n)` kjører:

```
permuter(arr, 3, n);                                  // Behold bare den som står i arr[2]
bytt(arr[2], arr[3]);  permuter(arr, 3, n);     // Bytt arr[3] ned i arr[2]
bytt(arr[2], arr[4]);  permuter(arr, 3, n);     // Bytt arr[4] ned i arr[2]
bytt(arr[2], arr[5]);  permuter(arr, 3, n);     // Bytt arr[5] ned i arr[2]
roterVenstre(arr, 2, n);                              // Venstreroter/rydd opp !
```