# Øving 2: Array-Liste som utvider seg i begge retninger (Array Deque)

## Godkjenning

Øvingen er obligatorisk. Øvingen godkjennes i øvingstimene av studentassistent med mindre noe annet er avtalt med faglærer. Dere kan gjøre og levere denne øvingen i grupper på to studenter.

## Læringsmål

Dere skal lære hvordan ei array-liste virker gjennom å implementere en variant av den. Dere skal lære hvordan implementere en enkel samling.

## Oppgave

Dere skal implementere ei array-liste som utvider seg i begge retninger. Denne versjonen av array-liste skal ha en egen metode appendleft for å legge til et element på starten av lista. Det skal settes av plass i starten slik at appendleft ikke trenger å flytte elementer inntil dette bufferet er oppbrukt.

Hvis appendleft kalles med tomt buffer i starten, skal lista utvides. Lag en ny array som er dobbelt så stor og kopier inn lista i den nye arrayen som for den eksisterende utvidelsesmetoden bortsett fra at den ekstra plassen skal settes av på starten. Det fører til at den gamle lista ikke skal kopieres inn på samme index i den nye, men skal kopieres inn seinere.

Dere skal bruke min array-liste implementasjon fra forelesningene og Canvas som basis for denne øvingen. For å sette av en buffer på starten trenger dere å lage en variabel som sier hvilken indeks i den underliggende array-en er indeks 0 i lista. Så må alle metodene i lista endres for å ta hensyn til at lista ikke starter på indeks 0 i arrayen.

**Obligatorisk:** Implementer eller modifiser konstruktøren samt følgende metoder. Konstruktøren kan godt ta størrelsen på startbufferet som parameter. Pop fjerner og returnerer siste element. Popleft fjerner og returnerer første element.

insert(index, element)

get(index) -> element

append(element)

appendleft(element)

pop() -> element

popleft() -> element

**Obligatorisk:** Analyser metodene du har implementert og oppgi kjøretida deres i O-notasjon.

**Frivillig:** Implementer __delitem__(index), __len__, __setitem__(index, item), reverse() samt implementer list slicing for de metodene som skal støtte dette. Implementer en iterator for lista.

**Frivillig:** Optimaliser insert() slik at den bruker kortest mulig tid. Optimaliseringen består i at hvis man setter inn nær starten så er det rakere å flytte de første få elementene mot starten og senke startbufferen enn å flytte nesten alle elementene mot slutten slik som basis-insert gjør.

**Frivillig:** Hvis elementer typisk legges til på slutten og fjernes på starten, vil lista etter hvert få mye tom plass på starten. Legg inn en sjekk for dette i popleft() og flytt lista i stedet for å øke bufferstørrelsen som normalt.

Merk: Lista fra denne oppgaven kan brukes som en kø eller en Deque. Deque er en forkortelse for «Double-ended Queue» eller dobbeltsidig kø på norsk. En dobbeltsidig kø er en kø hvor man kan sette inn og ta ut fra både starten og slutten.

## Illustrasjonseksempel

Eksempel: Starttilstand for ei liste med startstørrelse 10 og startbuffer 3. Merk at i dette tilfellet er indeksene i den underliggende arrayen annerledes enn indeksene som brukeren ser. I den underliggende arrayen er indeks 0 indeksen til elementet lengst til venstre, mens elementet som for brukeren har indeks 0 har indeks 3 i den underliggende arrayen. De tre elementene uten indeks for brukeren er startbufferen.

| **Indeks brukeren ser** | | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Indeks i arrayen** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Verdi** | - | - | - | - | - | - | - | - | - | - |

Eksempel: Tilstand etter kall av append(2), append (5), append (7)

| **Indeks brukeren ser** | | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Indeks i arrayen** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Verdi** | - | - | - | 2 | 5 | 7 | - | - | - | - |

Eksempel: Tilstand etter kall av appendleft(9)

| **Indeks brukeren ser** | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Indeks i arrayen** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Verdi** | - | - | 9 | 2 | 5 | 7 | - | - | - | - |

Eksempel: tilstand etter kall av appendleft (4) og appendleft (12). Startbufferet er nå tomt, så indeksen brukeren ser er den samme som indeksen i den underliggende arrayen.

| **Indeks brukeren ser** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Indeks i arrayen** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Verdi** | 12 | 4 | 9 | 2 | 5 | 7 | - | - | - | - |

Eksempel: Tilstand etter kall av appendleft (15), etter utvidelse i starten. Ved utvidelse i starten settes startbuffer lik så lang som lista ble utvidet med. Så den er 10 etter utvidelsen og 9 etter at det nye elementet (15) er satt inn.

| **Indeks Bruker** | | | | | | | | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Indeks array** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 51 | 16 | 17 | 18 | 19 |
| **Verdi** | - | - | - | - | - | - | - | - | - | 15 | 12 | 4 | 9 | 2 | 5 | 7 | - | - | - | - |