**Oppgave 1**

**a)**

Procedure push\_back(x):

array.append(x)

Procedure push\_front(x):

array.insert(0, x)

Procedure push\_middle(x):

lengde = |array|

array.insert((lengde+1)/2, x)

Procedure get(i):

print(array[i])

**c)**

push\_back vil kjøre ved O(1) konstant kjøretid da tiden for å legge til elementer i slutten av ett array er konstant.Dette vil da ta O(N) tid i programmet vårt hvor N er antall operasjoner.

push\_front vil kjøre ved O(l) hvor l er antall elementer i arrayet vårt. Dette vil da ta O(l\*N) tid i programmet vårt hvor N er antall operasjoner

push\_middle vil kjøre ved O(n) hvor n er antall elementer i arrayet vårt. Dette vil da ta O(l\*N) tid i programmet vårt hvor N er antall operasjoner.

get vil kjøre ved O(1) konstant kjøretid da kjøretiden til index oppslag i array er konstant. Dette vil da ta O(N) tid i programmet vårt hvor N er antall operasjoner.

**d)**

Kompleksiteten vil ikke endre seg, den vil fortsatt være avhengig av N. Men vi vil ha en øvre grense på hvor lang tid programmet kan ta.

**Oppgave 2**

Operasjonene inne i while løkken til algoritmen vil ta O(N) tid, fordi hvis man vil hente elementer i en lenket liste med indexen, vil man måtte traversere igjennon alle nodene til man kommer til ønsket index. Dette vil i værste fall være siste node, og kjøretiden blir derfor O(N). (N er antall elementer i den lenkede listen). Selv om vi teknisk sett ikke vil kunne lete etter siste elementet hver gang, og kjøretiden vil utvikle seg fra O(0.5N) - O(0.75N) - O(0.875) .. gradvis mot O(N), bryr vi oss ikke om konstantene og kjøretiden blir derfor O(N) while løkken vil kjøre log(N) ganger(binærsøk), og den totale kjøretiden blir derfor O(N \* log(N))

**Oppgave 3**

**a)**

Procedure lesFraFil:

HashMap

kattunge = input()

current = input()

while current != "-1":

delt = current.split()

forelder = delt[0]

for i in range(1, |delt|):

barn = delt[i]

map[barn] = forelder

current = input()

return stiFinner(kattunge, HashMap)

Procedure stiFinner(kattunge, HashMap):

strenge = str(kattunge)

current = map[kattunge]

while current in map:

strenge +=" " + str(current)

current = map[current]

return strenge

**Oppgave 4**

**a)**

Input: En liste av heltall i sortert rekkefølge

Procedure RekursivPrint(liste)

if ((|liste| -1) / 2)

end

i = ((|liste| -1) / 2)

MidtElement = liste[i]

VenstreSidenAvLista = liste[0:i]

HøyreSidenAvLista = liste[i+1:|liste|]

print(MidtElement)

RekursivPrint(VenstreSidenAvLista)

RekursivPrint(HøyreSidenAvLista)

**b)**

Input: En liste av heltall i sortert rekkefølge

Procedure RekursivHeapqPrint(heapq)

if (|liste| < 1)

end

KøVenstre

KøHøyre

MidtIndeks = (|heapq|/2)

for i in range(|heapq|)

if i < MidtIndeks

minsteElementIHeapq = heapq.pop

KøVenstre.push(minsteElementIHeapq)

elif i > MidtIndeks

minsteElementIHeapq = heapq.pop

KøHøyre.push(minsteElementIHeapq)

else

print(heapq.pop)

RekursivHeapqPrint(KøVenstre)

RekursivHeapqPrint(KøHøyre)