Oppgave 1a:

```
ArrayList<Integer> al
Algorithm push_front(tall)
   al.add(tall) på indeks 0

Algorithm push_middle(tall)
    al.add(tall) på indeks (al.size() + 1)/2

Algorithm push_back(tall)
    al.add(tall) (på indeks al.size())

Algorithm get(indeks)
    print(al(indeks))

```

Oppgave 1c:

push_front()
→ O(n) worst case, fordi alle elementene må i worst case kopieres til et nytt array

push_middle()
→ O(n) worst case, fordi alle elementene må i worst case kopieres til et nytt array

push_back()
→ O(n) worst case, fordi alle elementene må I worst case kopieres til et nytt array

get()
→ O(1) alltid lineært, fordi arraylist støtter direkte oppslag på indeks

Oppgave 1d:

Hvis vi vet at N er begrenset, betyr det at algoritmen kjøres i konstant tid.

Oppgave 2:

I et verste- tilfelle- estimat, når x ikke er i lista, vil algoritmen gi O(log(n)) fra binærsøket, og O(n) fra get- operasjonen til lenkede lister. Altså O(n*log(n)). Hvis get- operasjonen hadde vært logaritmisk/ konstant, kunne algoritmen spart tid.

Oppgave 3a:

```
Input: Strenger med tall, hvor første tall er forelder til de neste.
Første input- streng er posisjon til en katt.
Utput: Streng med tall som viser vei fra katt til det tallet som ikke har forelder
int streng <- USERINPUT
Map tre = Map<int,int>
Procedure FindWay(foreldre,barn)
    int katt = streng
    tre <- (katt,null)
    while (not streng = "-1")
        int forelder <- string[0]
        for i <- 1 to streng.lengde do
            int barn <- streng[i]
            tre <- (forelder,barn)
        Endfor
    Endwhile
    int value = 0
    while (true)
        if not map.get(number) = null do
            int value <- map.get(number)
            number = value
            Print(number)
        Else Endwhile
    Endwhile
```

Oppgave 4a:

```
Input: Streng med tall sortert fra minst til størst
Output: Streng med tall som lager et balansert binært søketre hvis brukt
til dette

streng <- userinput
arrayList
procedure toArrayList(streng)
    for i <- 0 to streng.lengde do
        arrayList.add(streng[i])

end procedure toArrayList

procedure utskrift(arrayList)
    v <- ArrayList
    h <- ArrayList
    if(arrayList.lengde = 0) do
        return
    end if

    else if(arrayList.lengde < 3)
        print alle elementer av arrayList
    end else if

    else if(arrayList.lengde > 2)
        mid = arrayList.lengde / 2
        for(i <- 0 to mid) do
            fjern første element fra arrayList og legg til på posisjon v(0)
        end for
        fjern første element fra arrayList og print dette
        h = arrayList
        utskrift(v)
        utskrift(h)
    end else if


end procedure utskrift
```

Oppgave 4b:

```
Input: Streng med tall sortert fra minst til størst
Output: Streng med tall som lager et balansert binært søketre hvis brukt
til dette

streng <- userinput
arrayList
procedure toPriorityQueue(streng)
    for i <- 0 to streng.lengde do
        pq.add(streng[i])

end procedure toArrayList

procedure utskrift(pq)
    v <- PriorityQueue
    h <- PriorityQueue
    if(pq.lengde = 0) do
        return
    end if
```

```
    else if(pq.lengde < 3)
        print alle elementer av pq
    end else if

    else if(pq.lengde > 2)
        mid = pq.lengde / 2
        for(i <- 0 to mid) do
            fjern første element fra pq og legg til på posisjon v(0)
        end for
        fjern første element fra pq og print dette
        h = pq
        utskrift(v)
        utskrift(h)
    end else if


end procedure utskrift
```