(1,2,3) ==> ((), (2), (3), (2,3), (1), (1,2),(1, 3), (1,2,3))

(2,3) ==> ((), (2), (3), (2,3))

5 ==> false

4 ==> true

3 ==> false

```
Algorithm isEven(n)
n       if n = 0 then
1               return true
n       return ! isEven(n-1)
```

power(x, 4) ==> x*x*x*x

power(x, 3) ==> x*x*x

```
Algorithm power(x, k)
        if k=0 then
                return 1
        return power(x, k-1) * x
```

```
Algorithm removeDups(L)
1       newL := new List
1       PQ := new PriorityQueue
1       p := L.first()
1       e := p.element()
1       PQ.insertItem(e, e)
n       while ! L.isLast(p) do
n               p := L.after(p)
n               e := p.element()
nlgn            PQ.insertItem(e, e)
n       while ! PQ.isEmpty() do
nlgn            e := PQ.removeMin()
n               newL.insertLast(e)
n               while !PQ.isEmpty() /\ e = PQ.minKey() do
nlgn                    PQ.removeMin()
1       return newL
nlog n
```

```
Algorithm insertionSort(L)
    if L.size() <= 1 then return
    p := L.first()
    while ! L.isLast(p) do
        p := L.after(p)
        q := p
        e := p.element()
        while !L.isFirst(q) /\ e < L.before(q).element() do
            q := L.before(q)
            L.replaceElement(L.after(q), q.element())
        L.replaceElement(q, e)
```