Given a sorted partition, to get the next sorted partition:
1. Cut off the final item.
2. Increase the sorted list of all sorted partitions of a given natural
   $n$.

Part (a) is subsumed by part (c).

Given a partition, to get the next partition:
1. Cut off the final item.
2. Increase the new final item by 1 and call this
   $i$.
3. Catenate 1s as necessary to make up the right sum (easily determined from
   the item that was cut off).

Let $L: [*(*(nat+1))]$ be a list-of-partitions variable whose final value is what we want. Then the problem is $R$, defined as

$$ R \quad = \quad (L' \text{ is the sorted list of all partitions of } n) $$

$$ \quad = \quad (\forall i, j: 0,..#L': i < j \Rightarrow L'i < L'j) $$

$$ \quad \land \quad (\forall i: 0,..#L': \sum L'i = n) $$

$$ \quad \land \quad (\forall Q: [*(*(nat+1))]; \sum Q = n \Rightarrow Q: (0,..#L')) $$

Introduce partition variable $P: [*(*(nat+1))$ and define

$$ A \quad = \quad (P \text{ is a partition}) \land (L \text{ is the sorted list of all partitions of } n \text{ that precede } P) $$

$$ \quad = \quad (\sum P) = n $$

$$ \quad \land \quad (\forall i, j: 0,..#L: i < j \Rightarrow Li < Lj < P) $$

$$ \quad \land \quad (\forall i: 0,..#L: \sum Li = n) $$

$$ \quad \land \quad (\forall Q: [*(*(nat+1))]; \sum Q = n \land Q < P \Rightarrow Q: (0,..#L)) $$

Now the refinements.

$$ R \quad \iff \quad A \Rightarrow R $$

$$ A \Rightarrow R \quad \iff \quad L := L^+[P]. $$

if #P < 2 then ok
else $P := P[0;..#P-2] + [P(#P-2)+1; (P(#P-1)-1)*1]$. $A \Rightarrow R$ fi

Here is a program using loops (Chapter 5); instead of gathering the partitions into a list, I print them (let's say !x prints the value of x and ?x reads into variable $x$).

```
var n, m: int; n is the length of P and m is a temporary
!"n=”. ?n.

var P: [n*int];
for i:= 0,..n do P i := 1 od.
for i:= 0,..n do !P i, “” od. !newline.
exit when n<2.
P(n-2) := P(n-2)+1.
m := n-1. n := m+P(m)-1.
for i:= m,..n do P i := 1 od od
```

The exact execution time is obtained by putting $t := t+1$ in front of the recursive call, and replace $R$ by $t' = t + 2^{n-1}-1$
replace $A \Rightarrow R$ by $t' = t + \Sigma i: 1,..#P. 2^{(\sum P[i;..#P])}-1$

the sorted list of all sorted partitions of a given natural $n$. For example, if $n=3$ then the answer is $[[1; 1; 1]; [1; 2]; [3]]$.

Given a sorted partition, to get the next sorted partition:
cut off the final item; increase the new final item by 1 and call this $f$; catenate as many $f$s as possible without making
the sum too big; increase the final item to get the right sum. This solution is very similar to part (c), but the assignment

\[ P := P[0;..#P-2] + [P(#P-2)+1 ; (P(#P-1)-1)*1] \]

has to be replaced by

\[ d := P(#P-2) + P(#P-1), f := P(#P-2)+1. \]

\[ P := P[0;..#P-2] + [(\text{div } d f - 1)*f ; f + \text{mod } d f] \]

and we have to modify \( R \) and \( A \).