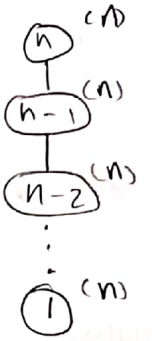
Question 1:

You are given 2 implementations for a recursive algorithm that calculates the sum of all the elements in a list (of integers):

```
def sum_lst1(lst):
  if (len(lst) == 1):
      return lst[0]
  else:
      rest = sum_lst1(lst[1:])
      sum = lst[0] + rest
      return sum
```



$= n^2$

```
def sum_lst2(lst, low, high):
  if (low == high):
      return lst[low]
  else:
      rest = sum_lst2(lst, low + 1, high)
      sum = lst[low] + rest
      return sum
```

Faster →

$= n$

Note: The implementations differ in the parameters we pass to these functions:

- In the first version we pass only the list (all the elements in the list have to be taken in to account for the result).
- In the second version, in addition to the list, we pass two indices: low and high (low ≤ high), which indicate the range of indices of the elements that should to be considered.
  The initial values (for the first call) passed to low and high would represent the range of the entire list.

1) Make sure you understand the recursive idea of each implementation.
2) Analyze the running time of the implementations above. For each version:
   i) Draw the recursion tree that represents the execution process of the function, and the local-cost of each call.
   ii) Conclude the total (asymptotic) running time of the function.
3) Which version is asymptotically faster?

Question 2:

Analyze the running time of each of the following functions. For each function:

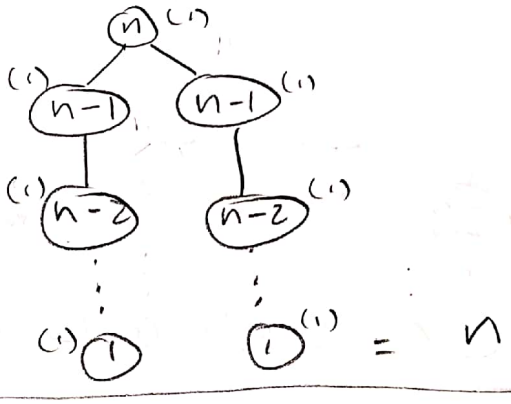
i. Draw the recursion tree that represents the execution process, and the cost of each call.

ii. Conclude the total (asymptotic) running time.

Note: For the simplicity of the analysis of sections (b) and (c), you may assume that n is a power of 2, therefore it can always be divided evenly by 2.

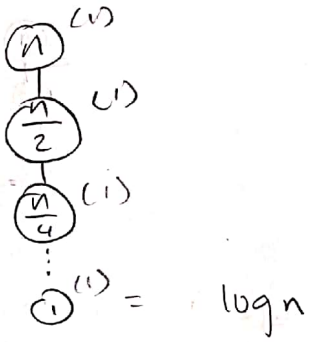
a.

```
def fun1(n):
    if (n == 0):
        return 1
    else:
        part1 = fun1(n-1)
        part2 = fun1(n-1)
        res = part1 + part2
        return res
```



b.

```
def fun2(n):
    if (n == 0):
        return 1
    else:
        res = fun2(n//2)
        res += n
        return res
```



c.

```
def fun3(n):
    if (n == 0):
        return 1
    else:
        res = fun3(n//2)
        for i in range(1, n+1):
            res += i
        return res
```