Orders of Growth

For each function below:

- Break it up into blocks, as we did in the chapter.
- Identify the order of growth for each block, or draw the recursive tree if applicable.
- Identify the order of growth for the whole function.

```
def min(lst):
    if not lst:
        return None
    min = lst[0]
    for elem in lst:
        if elem < min:
            min = elem
    return min

def captains_log(n):
    while n > 0:
        j = 1
        while j < n:
            j *= 2
            print("Captain's log")
        n -= 1

def turbo_count(n):
    for i in range(42000):
        print(i)

def blink(n):
    i = 1
    while i < n ** 2:
        for j in range(i):
            print()
        i *= 2

def layer_cake(n):
    if n > 1:
        layer_cake(n / 2)
        layer_cake(n / 2)
```

```
def fib_iterative(n):
    curr, next = 0, 1
    while n > 0:
        curr, next = next, curr + next
        n -= 1
    return curr

def fib_recursive(n):
    if n <= 1:
        return n
    return fib_recursive(n-1) + fib_recursive(n-2)
```

Code Writing

Write a program called `factors` that takes in a number `n`, and returns a set containing all the factors of `n`. It should run in $\Theta(\sqrt{n})$ at most.

```
from math import sqrt
def factors(n):
```