You are given an array of `logs`. Each log is a space-delimited string of words, where the first word is the **identifier**.

letter / digit

Easy

There are two types of logs:

- **Letter-logs**: All words (except the identifier) consist of lowercase English letters.
- **Digit-logs**: All words (except the identifier) consist of digits.

Reorder these logs so that:

1. The **letter-logs** come before all **digit-logs**.
2. The **letter-logs** are sorted lexicographically by their contents. If their contents are the same, then sort them lexicographically by their identifiers.
3. The **digit-logs** maintain their relative ordering.

1-st

2-St

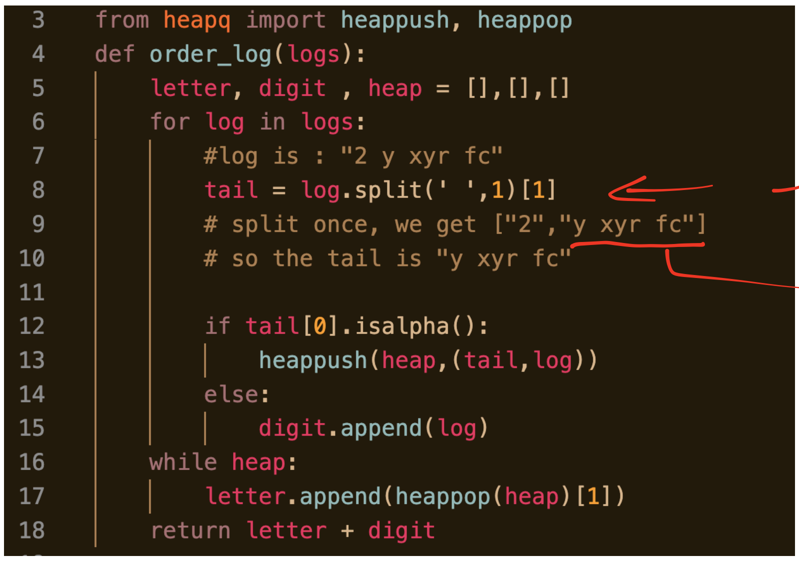
Return *the final order of the logs*.

Example 1:

```
Input: logs = ["dig1 8 1 5 1","let1 art can","dig2 3 6","let2 own kit dig","let3 art zero"]
Output: ["let1 art can","let3 art zero","let2 own kit dig","dig1 8 1 5 1","dig2 3 6"]
Explanation:
The letter-log contents are all different, so their ordering is "art can", "art zero", "own kit dig".
The digit-logs have a relative order of "dig1 8 1 5 1", "dig2 3 6".
```

A. Use heap to maintain the lex-order.

push (content , log) onto heap

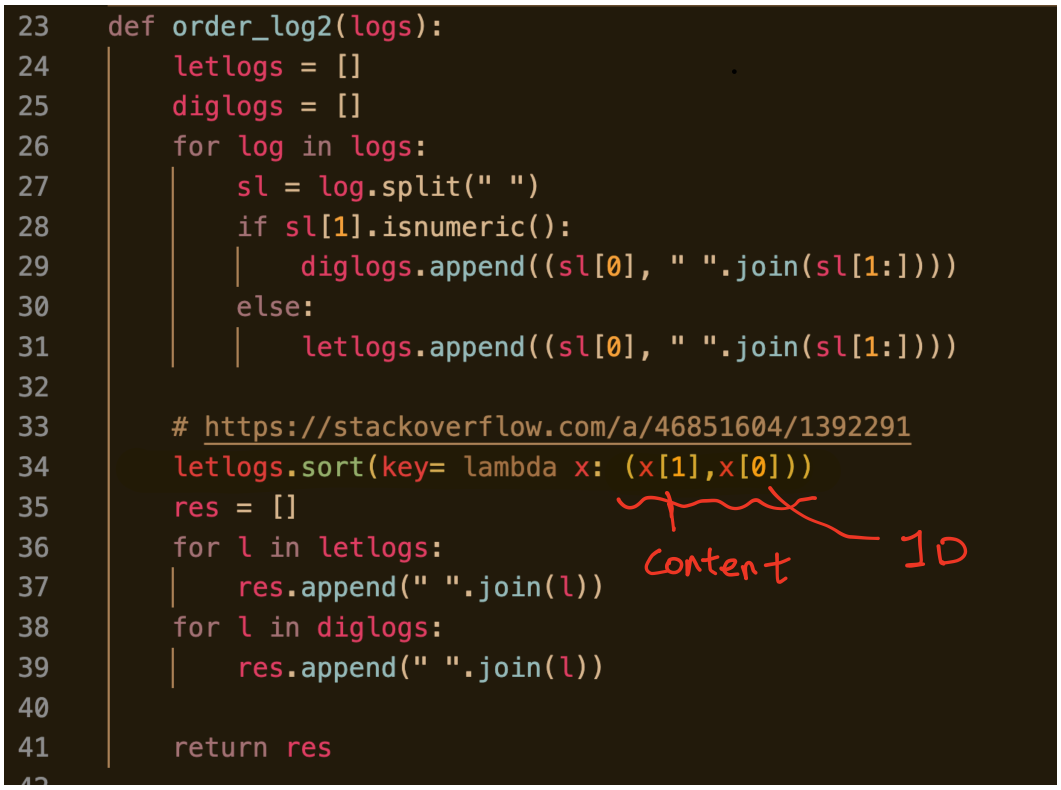


```python
from heapq import heappush, heappop
def order_log(logs):
    letter, digit , heap = [],[],[]
    for log in logs:
        #log is : "2 y xyr fc"
        tail = log.split(' ',1)[1]
        # split once, we get ["2","y xyr fc"]
        # so the tail is "y xyr fc"

        if tail[0].isalpha():
            heappush(heap,(tail,log))
        else:
            digit.append(log)
    while heap:
        letter.append(heappop(heap)[1])
    return letter + digit
```

[illegible] ID | content (tail)

B. use .sort( ) with key = lambda



```python
def order_log2(logs):
    letlogs = []
    diglogs = []
    for log in logs:
        sl = log.split(" ")
        if sl[1].isnumeric():
            diglogs.append((sl[0], " ".join(sl[1:])))
        else:
            letlogs.append((sl[0], " ".join(sl[1:])))

    # https://stackoverflow.com/a/46851604/1392291
    letlogs.sort(key= lambda x: (x[1],x[0]))
    res = []
    for l in letlogs:
        res.append(" ".join(l))
    for l in diglogs:
        res.append(" ".join(l))

    return res
```

考察知识点 Sort with

2 attribute, content 1st

then ID.