General Note:

- Read and understand the question carefully.
- For implementation, you are permitted to use the code that you have submitted as an assignment.
- You are not permitted to use global variables and/or static variables, unless specified.
- Assume that all the inputs in the test cases are valid.

1. On Rima's birthday, her friends hid a gift at her home and decided to play a game to find the gift's location. They lined up some boxes in the room where the birthday party is taking place. All but some of the boxes contained a few pieces of paper. Each piece has a letter and a number written on it. Some of the letters from a box forms a word. In a box, the letters present in the word are marked with unique positive integers, and those not present are marked with 0. The same letter can appear multiple times in the same box, but with different numbers.

   Rima has to find the location (which is a sentence) of the gift by arranging the words hidden in the boxes in the order in which the boxes are placed. The word hidden in a box can be obtained by assembling the letters in that box in the increasing order of the associated numbers.

   Given the letter-number pairs in each box in the order of boxes, your job is to help Rima in constructing the location name by performing the following tasks:

   - Find the word present in each box using a priority queue implemented as a heap (using array of structures).
   - Concatenate the words from all boxes to find the location of the gift.

   Use the following structure to store a letter and its associated number.

```
struct SLIP
{
  char letter;
  int number;
};
```

   Use the following global variable to store the decoded location.

```
char location[10000];
```

   Your program should implement the following functions using priority queue as per the given function prototypes.

   - $main()$: Repeatedly read a character 'r', 's', or 'd' from the console and call the corresponding functions, as described below, until character 't' is encountered. [2 Marks]
   - $insert\_queue(Q, slip, n)$: Insert $slip$ into the priority queue Q with current length n by using the concept of insertion into a heap as mentioned in CLRS, without storing it into another array. After the insertion, print the letters in Q in the increasing order of their position in the array, separated by a space. [2 Marks]
   - $decode\_word(Q, n)$: If the queue is not empty, decode the word of length n from the priority queue Q using the extract function of heap, print it, and append it to $location$. Otherwise, print -1. [2 Marks]

Input/Output Format

The input consists of multiple lines. Each line starts with a character from $\{r, s, d, t\}$ followed by zero or one character and integer.

- Character 'r' : Character 'r' will be followed by a character $c \in [a - z, A - Z]$ and a non-negative integer $i \in [0, 10^5]$ representing the letter and associated number, respectively. Read c and i as a *slip* (variable of `struct SLIP`) and if i is not equal to 0, call function $insert\_queue()$.
- Character 's' : Character 's' represents that the letters from the current box are finished. Call function $decode\_word()$.
- Character 'd' : Display $location$.
- Character 't' : Terminate the program.

Input 1

```
r o 250
r g 101
r l 257
r d 1009
r f 0
s
r o 888
r x 7003
r b 765
s
d
t
```

Output 1

```
o
g o
g o l
g o l d
gold
o
o x
b x o
box
gold box
```

Input 2

```
r r 5000
r e 500
r u 1
r e 0
r d 256
r n 222
s
```

```
s
r n 1777
r i 1212
r i 251
r n 1111
r d 12
r g 4561
s
d
r t 2
r b 32
r a 12
r l 33
r e 35
s
d
t
```

Output 2

```
r
e r
u r e
u d e r
u n e r d
under
-1
n
i n
i n i
i n i n
d i i n n
d i i n n g
dining
under dining
t
t b
t b a
t b a l
t b a l e
table
under dining table
```