Binary Tree Opeartions

Given different sets of inputs perform various operations on a Binary Tree. Every TreeNode contains a pointer to left child , a pointer to right child, A character named "name" and an integer named "val". Here Are the descriptions of various operations

```
struct TreeNode
{
    TreeNode *left;
    TreeNode *right;
    char name;
    int val;
}
```

1. Insertion (Operation number 1) : Given n integers followed by n pairs of space separated integers and characters , create a Tree rooted a particular Node. IF there is an integer value "-1" in the inputs then leave that position blank.Every "-1" is followed by a "N" (All other valid Nodes have lower case char as name).

Where you should Insert?

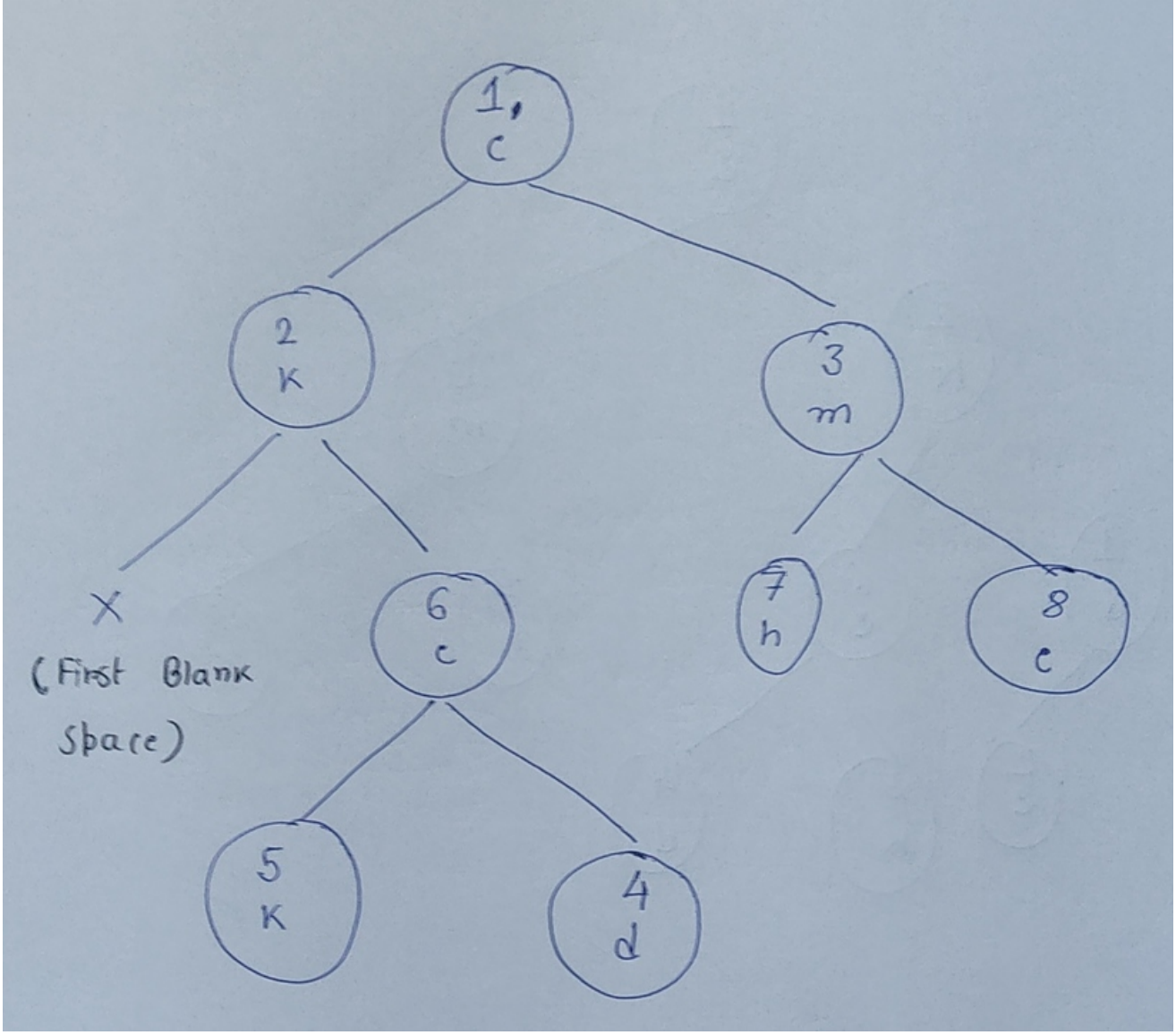
If Tree is blank, then create a Tree at "root" only. Else Find first Blank position. And add the newly created Tree at this postion.

What is First Blank Position?

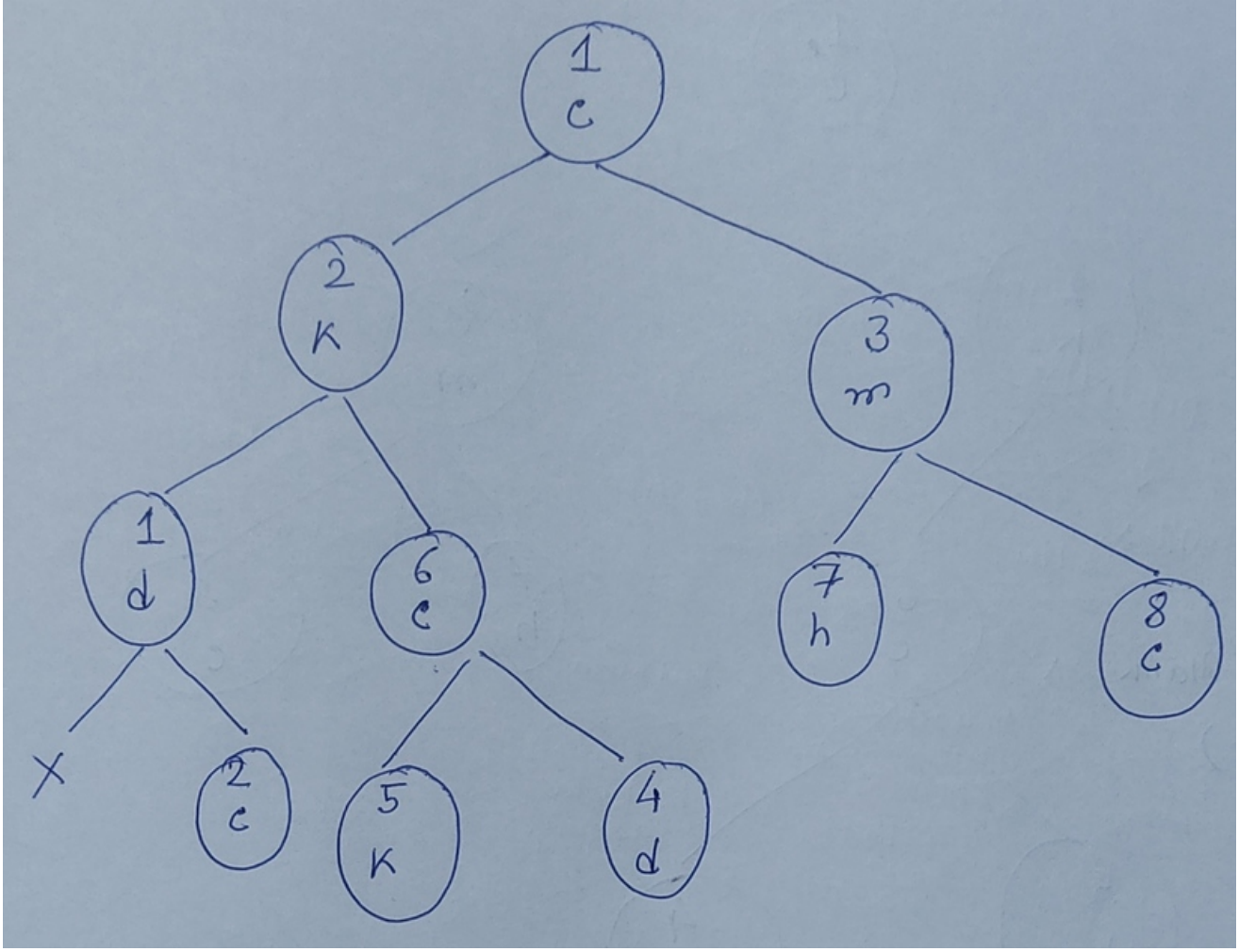
Traverse the tree level wise left to right, whenever you get any empty Node poistion for the first time, This should be the First Blank position.

Here is an example: (First Insertion, Tree is empty now)

n=7 1 c 2 k 3 m -1 N 6 c 7 h 8 c 5 k 4 d



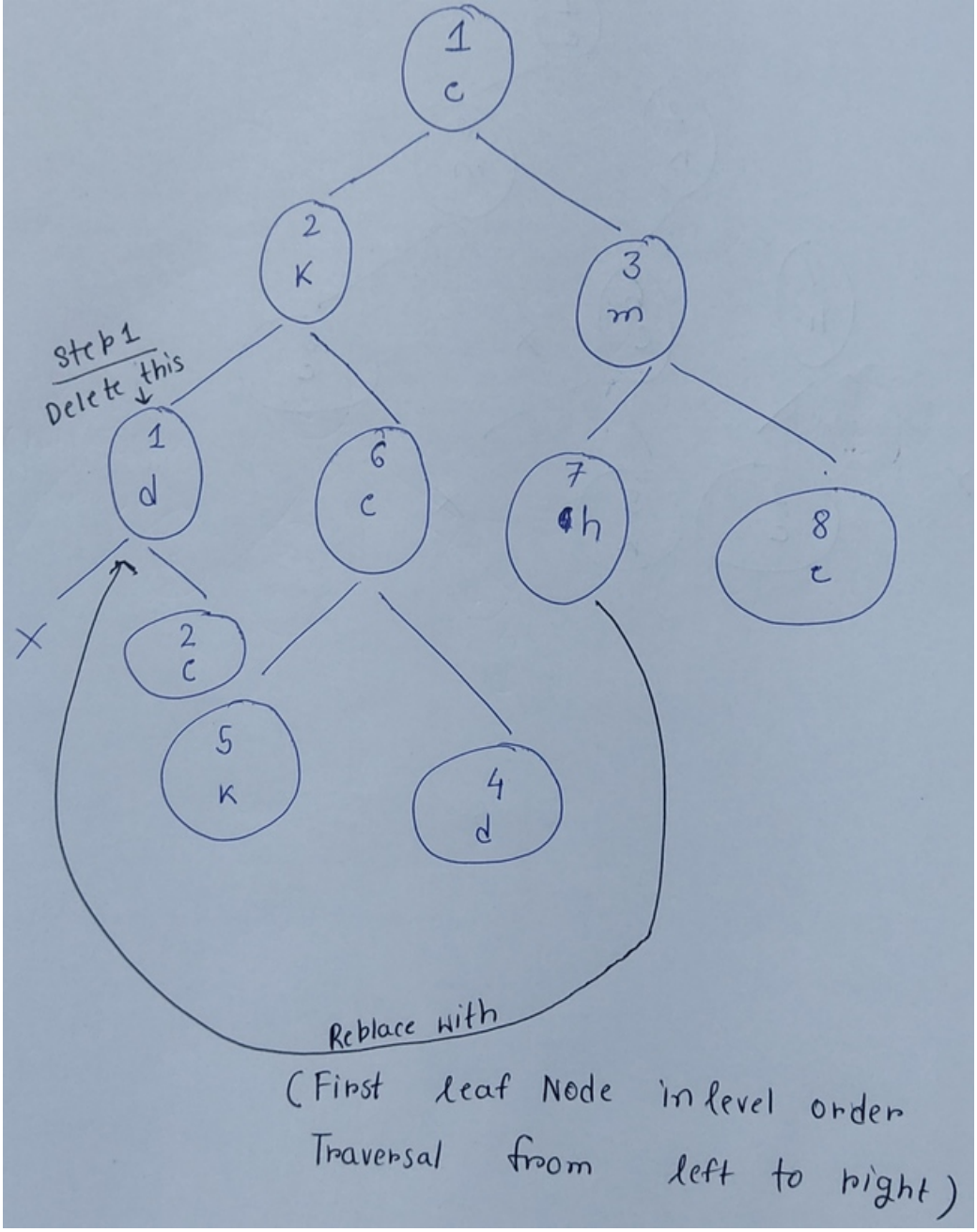
Now, if we again Insert with n=3 1 d -1 N 2 c Tree will look like



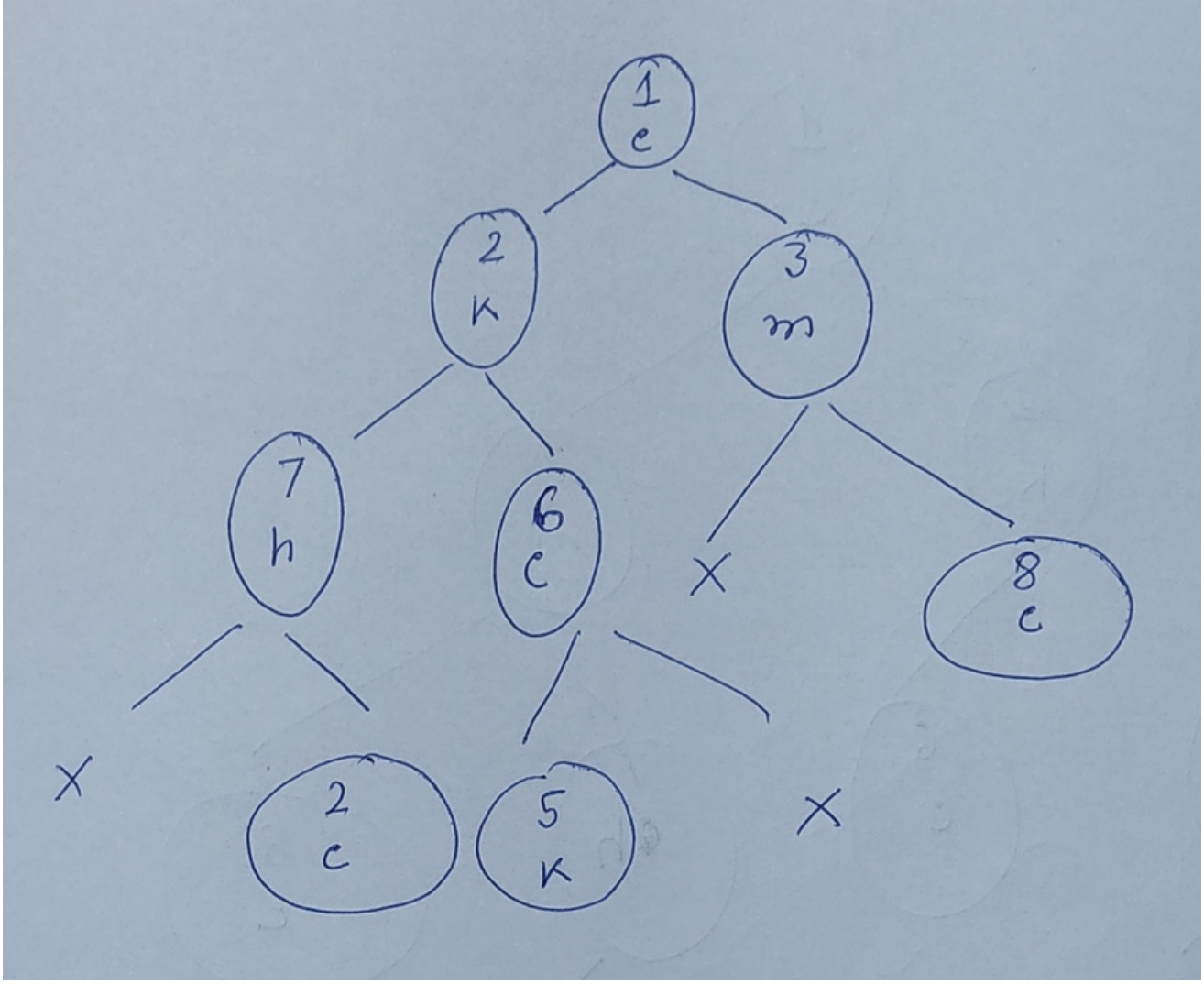
2. Deletion (Operation number 2): Given a "Target" string ,Traverse the Tree level by level from left to right and Delete All the nodes where TreeNode->name matches "Target".After Deletion , if the deleted Node is non-leaf then replace this Node with a leaf Node where "name" doesn't match with "Target".

Which Leaf Node to replace with? Traverse the Tree level wise, and replace with the first leaf Node you encounter.

Suppose "Target"=d



After Deletion Final Tree will look like this



3. Serach (operation number 3): Given a "Target" string print how many times "Target" appears in the entire Tree.

Ex: Target = 'c' output = 4

4. Find Leaf and non-leaf (operation number 4): Find number of leaf and non leaf Nodes where "val" is equal to a given integer value "Target"

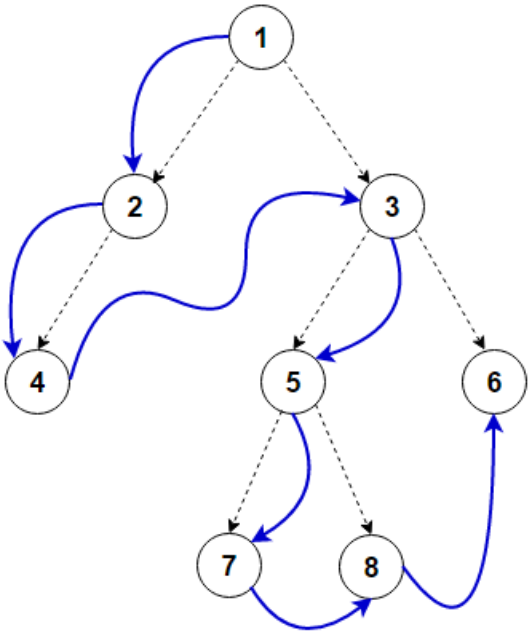
Target = 4 output: 1 1

Target = 3 output: 0 1

5. Print : Print all the Nodes of the Tree in preorder fashion where TreeNode->val is greater than the given "Target" value. (After every printing every Node name give a single space)

What is preorder?

Preorder Traversal is a way of Traversing a Tree where we first explore root, then left child and then right child i.e. (root,left,right) in this order. Below is an example of preorder Traversal.



Preorder: 1, 2, 4, 3, 5, 7, 8, 6

Ex: Target=0

output: c k h c c k m c

Input Format

for every test cases, the input format contains the opration number and the required inputs for that operation. First It will have total number of operations 'T' followed by T number of operations as follows:

1. 1 For Insertion , followed by an integer n , followed by n pairs of space separated Integer-character pair.
2. 2 For Deletion , followed by a single character "Target"
3. 3 for Searching , followed by a single character "Target"
4. 4 for Finding leaf and non-leaf, followed by a single integer "Target"
5. 5 For printing followed by an integer "Target"

Constraints

1<=opertion<=5 1<=n<=100 1<=TreeNode->val<=9

Output Format

For Insertion and Deletion do nothing. For operation 3, print the result and go to next line. For operation 4, print leaf_node_count followed by space and non_leaf_count and go to next line. For operation 5, print TreeNode->name and give a blank space after every name, and at the end print a new line

For simplicity we are providing a starter code for I/O

```
int main()
{
    int T;
    cin>>T;
```

```
    struct Node *root=NULL;
    while(T--)
    {
        int operation;
        cin>>operation;
        if (operation==1)
        {
            int n;
            cin>>n;
            vector <int> vals(n,-1);
            vector <char> names;
            for (int i=0;i<n;i++)
            {
                char s;
                int v;
                cin>>v;
                cin>>s;
                vals[i]=v;
                names.push_back(s);
            }
            if (root != NULL) //Tree already exists
            {
               //your code goes here
            }
            else
            {
            //your code goes here
            }
        }
        if (operation==2)
        {
            char s;
            cin>>s;
            //your code goes here
        if (operation==3)
        {
            char s;
            cin>>s;
            int count=0;
           //your code goes here
            cout<<count<<endl;
        }
        if (operation==4)
        {
            int target;
            cin>>target;
            int leaf=0,non_leaf=0;
            //your code goes here
            cout<<leaf<<" "<<non_leaf<<endl;
        }
        if (operation==5)
        {
            int target;
            cin>>target;
            //your code goes here
            cout<<endl;
        }
    }

    return 0;
}
```

Sample Input 0

```
7
1
9
1 c
```

```
2 k
3 m
-1 N
6 c
7 h
8 c
5 k
4 d
1
3
1 d
-1 N
2 c
2
d
3 c
4
2
4
3
5
0
```

Sample Output 0

```
4
1 1
0 1
c k h c c k m c
```

Sample Input 1

```
9
1
15
2       d
7       i
1       r
3       h
6       n
4       z
5       q
6       j
3       w
-1       N
1       j
9       s
8       s
3       o
6       g
5
1
3
o
2
k
2
f
5
7
3
y
5
5
5
8
```

Sample Output 1

```
d i h j w n z s s q o g
1
s s
0
i j n s s g
s
```

Sample Input 2

```
29
1
84
22      p
24      w
26      e
9       d
21      k
22      b
7       r
9       w
42      y
37      g
5       b
16      y
44      d
34      l
37      b
21      c
14      e
21      i
38      w
33      d
41      x
19      k
32      c
37      f
16      t
44      a
17      r
33      t
10      h
37      g
36      h
29      x
20      f
31      l
12      v
12      i
28      m
15      p
29      r
36      k
31      i
20      q
28      c
8       w
36      a
44      k
42      g
6       f
14      f
31      y
21      e
```

42 n
38 m
26 v
6 c
33 h
7 p
30 k
44 b
38 z
8 n
6 h
16 m
24 o
6 j
13 i
31 z
34 h
23 u
13 g
44 q
34 j
32 s
9 i
43 n
18 l
8 n
44 u
22 b
42 s
35 s
24 z
20 m
44 c
3
t
2
e
5
10
4
6
2
s
4
25
1
94
29 r
27 x
14 i
13 j
37 z
12 d
40 e
35 e
33 h
7 m
26 k
27 r
33 m
22 s
44 r
42 q
22 r
9 c
43 t
43 y
40 r
27 j
7 y
35 y

17 g
13 q
12 h
28 j
8 m
42 q
12 o
27 p
22 o
9 m
27 e
18 b
36 o
7 c
34 n
6 m
11 u
27 a
14 h
10 r
19 t
39 s
30 q
18 p
5 i
21 p
24 e
8 d
33 p
16 e
28 s
19 m
8 o
23 x
6 d
35 u
35 n
14 r
41 f
23 g
42 w
36 k
7 n
31 q
14 p
22 o
18 b
42 m
16 s
42 z
7 q
25 a
39 g
29 j
30 o
18 e
22 v
22 h
19 e
30 u
35 f
33 x
39 t
27 z
42 r
5 y
23 m
27 j
16 y
30 s
4

10
3
d
3
b
2
s
4
13
3
g
3
v
5
5
1
77
38 n
22 o
23 k
15 x
17 y
37 d
32 z
28 z
26 p
17 p
36 y
20 h
36 m
8 x
14 h
41 z
19 a
37 d
7 w
30 q
10 j
39 j
42 a
29 j
15 u
24 h
39 p
42 o
34 q
23 w
9 k
38 i
26 d
15 p
7 l
21 o
16 n
6 q
22 h
24 j
42 a
38 i
37 v
35 h
17 s
39 d
8 h
8 e
20 u
35 j
6 h
27 c
10 y
33 d

```
19      l
12      a
28      v
11      t
7       b
25      b
16      z
7       u
20      s
17      g
15      k
18      x
44      y
41      v
9       a
16      p
17      w
26      o
19      q
6       v
19      i
33      d
32      a
5
19
5
14
2
r
2
1
5
30
5
17
3
d
2
u
5
16
3
s
3
e
5
20
```

Sample Output 2

```
2
p w c x o f i z l h u v g q y i i j s m n w p l r u b k g d k s s i z m x q c k a c k g c b y f f t y d a n m
r v l t h k b b g z h m
4 0
0 0
0 2
6
7
2 2
7
3
p w d w c x o j f i z w l h u v g q y i i j m i n w p l n r u b k g d i z m x q c r x j e q p g w o k n r m q
p e o b h c b m o z q t c a g n j o z m y m e v u h e r a u f h x t k j r z r t m y a j y q i d r y p g p e m
q d p h e e k j m o m x d r q u n o r f k c k g c b y f f f t y d a n m r v c r l t h p h k b b g z n h h m
p w c x o f z l h u q y i j m n w r u b k g d n o z z i d y v p d o o d a h q j a i v y j h a d k d h j u j m
h c p d z o v q w b s i z m x q c r x e q p g w o k r q e o h m o z t a g n j o z y v h r a u f x t k j z r m
a j q r y p e m p e k j x r q u n f c k g c b f y d a n m v l t h k b b g z h
p w c x o f z l h u q y i j m n w p l r u b k g d n o x z z i g k d x y a p v p w p d o o q n i d a h y p q j
```

a i v y j h s a d k d h j u u j m h c p d l z o v q w b z s i z m x q c r x e q p g w o k r q e o b h b m o z
t a g n j o z y e v h e r a u f x t k j z r t m a j y q r y p g p e m p e e k j m x r q u n f k c k g c b y f
t y d a n m r v l t h k b b g z h m
z h q y j n w g u g d n z i y v n d d a a i v y j h a d d j m p d z o q i x c c e q w k q h m o z t g n z y f
f x t y a y m p e k q u n f d a m k t h b b g z h
p w c x o f z h u q y i j m n w g u b k g d n o z z i d x y a v n p d o o q i d a h q j a i v y j h a d k d h
j u j m h c p d z o v q w b s i z m x q c c x e q p g w o k q e o b h b m o z t a g n j o z y e v h e f a u f
x t k j y z t m a j q y p p e m p e k j m x q u n f c b d a m k v k t h k b b g z h
12
p w c x o f z h q y i j m n w g b k g d n o z z i g d x y a v n w p d o o q i d a h y p q j a i v y j h s a d
k d h j m j m h c p d z o v q w b s i z m x q c c x e q p g w o k q e o b h b m o z t a g n j o z y e v a h e
f a f x t k j y z t m a j q y p g p e m p e k j m x q n f c b d k v k t h k b b g z h
2
8
p w c x o z h q y i j m n w g b k g d n o z z i d y v n p d o o d a h q j a i v y j h a d k d j m j m h c p d
z o v q w b i z x c c x e q p g w o k q e o h m o z t a g n j o z y v a h f a f x t k j y z m a j q y p e m p
e k j x q n f c b d k v t h k b b g z h