Answer to the question no-01

```
s=input()
a=0
b=0
c=0
st = []
r=""
for i in s:
    if i=='L':
        a+=1
        r+=i
    elif i=='R':
        b+=1
        r+=i
    if a==b:
        st.append(r)
        c+=1
        r=""
        a=0
        b=0
print(len(st))
for i in st:
    print(i)
```

Answer to the question no-02

```
from collections import Counter
n=int(input())
A=list(map(int,input().split()))
num=Counter(A)
cnt=0
for i in num:
    if num[i]>i:
        cnt+=abs(i-num[i])
    elif num[i]<i:
        cnt+=num[i]
print(cnt)
```

Answer to the question no-03

a)Ans:

List	Dictionary
1. An ordered collection	1. An unordered collection
2. Access by their index	2. Accessed by their keys.
3. Mutable	3. Immutable
4. Can contain duplicate element	4. Contain only unique elements
5. When removing value it takes O(n) time complexity	5. It takes O(1) time complexity.

b)Ans:

*args:
The *args syntax allows a function to accept a variable number of positional arguments. The * (asterisk) unpacks the arguments passed to the function into a tuple, which can then be accessed within the function body.

```
def fun(*args):
    for arg in args:
        print(arg)
fun(1, 2, 3, 4)
output->
1
2
3
4
```

**kwargs:
The **kwargs syntax allows a function to accept a variable number of keyword arguments. The ** (double asterisks) unpacks the keyword arguments into a dictionary, which can then be accessed within the function body.

```
def fun(**kwargs):
```

```
    for key, value in kwargs.items():
        print(key, value)
fun(name='Mr.X', age=25, city='Feni')
```

output->
name Mr.X
age 25
city Feni

Using *args and **kwargs provides flexibility when defining functions that can handle a varying number of arguments, making code more efficient.

Answer to the question no-04

```
def odd(A):
    n=0
    for i in A:
        if i%2==0:
            n+=1
    return n

n=int(input())
A=list(map(int,input().split()))
a=0
while True:
    x=odd(A)
    if x==n:
        a+=1
        for i in range(0,n):
            A[i]=A[i]/2
    else:
        break
print(a)
```

Answer to the question no-05

```
import pyautogui
from time import sleep
n=int(input())
```

```
sleep(5)
for i in range(1,n+1):
    for j in range(0,i):
        pyautogui.write('#' , interval=0.25)
    pyautogui.press('enter')
```