**Practical 7**

**Aim:** Lapindrome is defined as a string which when split in the middle, gives two halves having the same characters and same frequency of each character. If there are odd number of characters in the string, we ignore the middle character and check for lapindrome. For example gaga is a lapindrome, since the two halves ga and ga have the same characters with same frequency. Also, abccab, rotor and xyzxy are a few examples of lapindromes. Note that abbaab is NOT a lapindrome. The two halves contain the same characters but their frequencies do not match. Your task is simple. Given a string, you need to tell if it is a lapindrome.

**Input:**

6

gaga

abcde

rotor

xyzxy

abbaab

ababc

**Output:**

YES

NO

YES

YES

NO

NO

s/w: PyCharm Community edition 2021.3

**Theory and approach:**

Our task is to check if the frequency of characters is the same on either half of the string. To achieve this task, we need to slice the string in such the way that the characters are equally distributed in both the halves.

// - is used for floor division. =>

Normal division using ‘/’ will not return whole number. To get the floor, we use this operator

word[0:len(word) // 2] =>

len(word) returns the number of characters in the word. We want to store the first half of the word variable ‘first’. So, we divide the length by 2 to get the first half of the word. Note that the last index in the slice is not included.

word[len(word) // 2+1:] or word[len(word) // 2:]=>

To store the second word, we need to check if the length is odd or even. If it is even, we directly store the remaining characters at the end. If it is odd, we need to skip the middle character and start storing from the next character till the end.

Later, convert both the string to list, sort them out and compare if they are equal. If the frequency is same, they’ll be identical and ‘YES’ is the output. If not, ‘NO’ is the output.

e.g.,

word = xyzxy

index: 0 1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | y | z | x | y |

len(word) = 5

first = word[0:len(word) // 2]

which is equivalent to, first = word[0:5//2] = word[0:2] = x y

(Note: index 2 is not included so z will not be stores)

Similarly, if since the length is odd, we need to skip the middle character ‘z’ and continue storing d=from the next character after z, to the end.

So, second = word[len(word) // 2 + 1:0]

Which is equivalent to word[5/2+1:] = word[2+1:] = word[3:] = x y

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | y | z | x | y |

first second

Upon storing into respective lists and then sorting using sort() which will sort the list alphabetically,

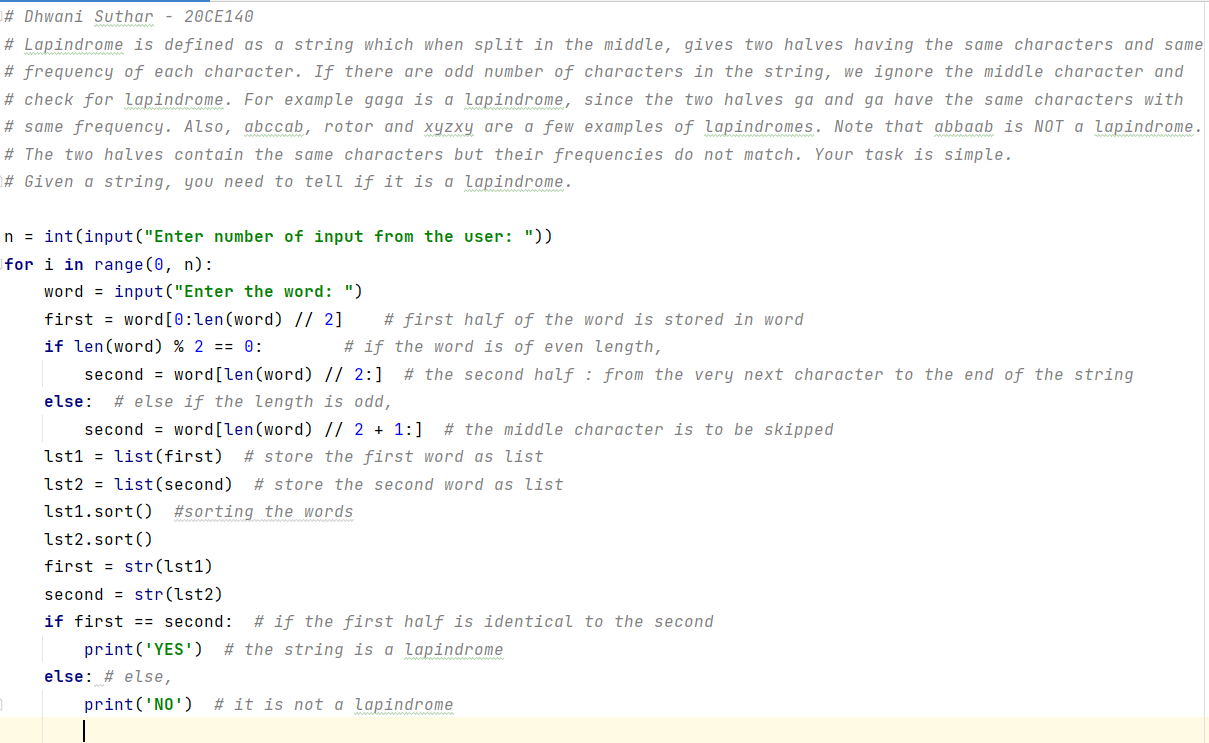
first = y z

second = y x

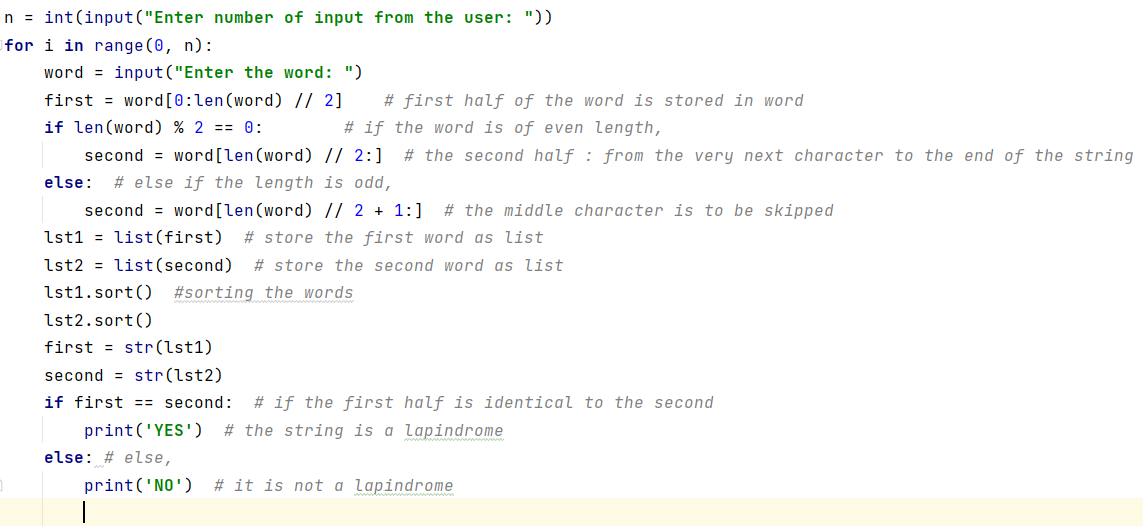
Lastly, we observe that first is identical to second. So, we know that the string is a lapindrome. Hence, we print ‘YES’

**Program:**

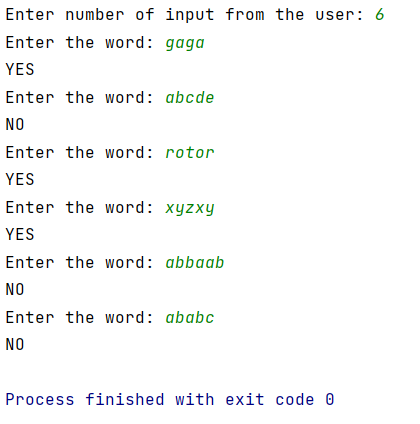
n = int(input(**"Enter number of input from the user: "**))  
**for** i **in** range(0, n):  
 word = input(**"Enter the word: "**)  
 first = word[0:len(word) // 2]  
 **if** len(word) % 2 == 0:  
 second = word[len(word) // 2:]  
 **else**:  
 second = word[len(word) // 2+1:]  
 lst1 = list(first)  
 lst2 = list(second)  
 lst1.sort()  
 lst2.sort()  
 first = str(lst1)  
 second = str(lst2)  
 **if** first == second:  
 print(**'YES'**)  
 **else**:  
 print(**'NO'**)



(zoomed)



**Output:**



**Conclusion:**

1. sort() : sorts the values alphabetically (in this case)

2. string[start:end+1] will slice the string from ‘start’ to ‘end’

3. // : floor division operator which divides and then converts the result to nearest whole number such that the whole number < result.

GitHub: