Week-4, Practice, Programming

Note: Check the comments in the code for the solution.

Problem 1

Question

Accept electricity `units` as a positive integer from the user and write a program to print total bill amount according to the following criteria:

Units	Cost per unit (`Rs`)
0 to 100	0
101 to 200	5
201 to 500	8
501 and above	10

Answer

```python
units = int(input())
bill = 0
#if units are greater than 500
if units > 500:
    bill += 5*100 + 300*8 + (units-500)*10
#if units are 201 to 500
elif units > 200:
    bill += 5*100 + (units-200)*8
#if units are 101 to 200
elif units > 100:
    bill += 5*(units-100)
#if units are 0 to 100
else:
    pass
print(bill)
```

Test Cases

Public

Input	Output
75	0
150	250
250	900

Private

Input	Output
0	0
200	500
300	1300
600	3900

Problem 2

Question

Write a program to accept a string from the user that contains `(`,`)`,`{`,`}` and `[`,`]` in it. Print `True` if all the brackets are opened and closed properly. Otherwise print `False`.

Note:

- `{}[]()` are the opening and closing brackets which needs to be verified - All the opening brackets should be closed with the same type of closing bracket.

Input	Output
`(jhdhd}(sdddd){)`	False
`a(h{g$2[j)h]h}`	False
`{abc(ddd)ee[ff()dd]ee}`	True

Answer

```python
s = input()
o = '({['    # Opening brackets
c = ')}]'    # Closing brackets for the opening bracket of same index
b = ''       # A string variable takes the open brackets
match = True # Boolean variable for validation
for i in s:
    if i in o:
        b = b+i # concatenated to b if it i a opening bracket
    if i in c:
        # o[c.index(i)] gives the matching opening bracket for the closing bracket i
        # o[c.index(i)] in b provides the presence of same type of bracket is opened before
        # o[c.index(i)] should be the last opened bracket to be closed.
        if o[c.index(i)] in b and o[c.index(i)] == b[-1]:
            b = b[:-1] # removing the last brack because it is properly closed
        else:
            match = False # validated to False if above conditions are not satisfied
            break
if len(b) != 0: # checking for remaining brackets which are not matched
    match = False
print(match)
```

Test Cases

Public

Input	Output
`(jhdhd}(sdddd){)`	False
`[{(sa]sa(aaa)}`	False
`[]{{}()}[{()}]`	True

Private

Input	Output
`a{kjjf(ddfffs)hh[f(hh)d]h}d(hhd)`	True
`[{{(([[jjhhh]]))}}]`	True
`[{{(([[jjhhh]}}]))]][`	False
`(({)})`	False

Problem 3

Question

Accept a string from the user and print the encrypted string according to the following conditions:-

- Each letter should be replaced by the letter which is at the same position from reverse in alphabets like `a` is replaced by `z`, `b` is replaced by `y`..... `y` is replaced by `b`, `z` is replaced by `a`
- Uppercase letters should be in uppercase and lowercase letters should be in lowercase after conversion.
- Each digit should be replaced by a digit which is at the same position from reverse in (0,1,2...9) like, `0` is replaced by `9`, `1` is replaced by `8` .....`8` is replaced by `1` and `9` is replaced by `0`.
- Blank space should be replaced by '_' and other types of character remain the same.

Answer

```python
#getting input
message1 = input()
alp = "abcdefghijklmnopqrstuvwxyz"
nm = "0123456789"
message2 = "";
# Read the each character from message1 one by one
for i in message1:
# If character is alphabet
    if i.isalpha() == True:
# If character is in uppercase
        if i.isupper() == True:
            index = alp.index(i.lower())
            message2 += (alp[25-index]).upper()
# If character is in lowercase
        else:
            index = alp.index(i)
            message2 += (alp[25-index])
# If character is digit
    elif i.isdigit() == True:
        index=nm.index(i)
        message2 += (nm[9-index])
# If character is blank space
    elif i == " ":
        message2 += "_"
# For other character
    else:
         message2 += i
print(message2)
```

Test Cases

Public

Input	Output
`abcde123`	zyxwv876
`This is Data Science course`	`Gsrh_rh_Wzgz_Hxrvmxv_xlfihv`
`abc@123.com`	`zyx@876.xln`

Private

Input	Output
`zyxwvutsrqp`	`abcdefghijk`
`@#&^*.()`	`@#&^*.()`
`ABCDEF@GHIJK`	`ZYXWVU@TSRQP`
`abcd efgh @ 9876543210`	`zyxw_vuts_@_0123456789`

Problem 4

Question

Accept a non-empty list of space-separated positive integers as input from the user and print all numbers in the list which are greater than the average in non-descending order. The output format should be a sequence of space-separated integers. For example:

Input

```
5 6 3 2 7 1 4 3
```

Output

```
4 5 6 7
```

Explanation

Average is $(5+6+3+2+7+1+4+3)/8 = 3.875$.

Answer

```python
# Getting input and after split from blank space assign to n
n=input().split(" ")
l=[]
total=0
# Append each number in l from n after convert str to int and calculate total
for i in n:
    l.append(int(i))
    total+=int(i)
# Sort the list elements
l.sort()
# Calculate average
average=total/len(n)
# Print each number which is greater than average
for i in range(len(l)):
    if l[i]>average and i!=len(l)-1:
        print(l[i],end=" ")
    elif l[i]>average and i==len(l)-1:
        print(l[i])
```

Testcases

Public

Input 1

```
9 8 7 6 5 4 3 2 1
```

Output 1

```
6 7 8 9
```

Input 2

```
2 2 2 2 2 2 2 3 3 3 3 3
```

Output 2

```
3 3 3 3 3
```

Input 3

```
5 5 5 5 6 6 6 6 4 4 4 4
```

Output 3

```
6 6 6 6
```

Private

Input 1

```
0 1 3 5 7 9 13 11 10 8 6 4 2
```

Output 1

```
7 8 9 10 11 13
```

Input 2

```
100 50 0 150 200
```

Output 2,

```
150 200
```

Input 3

```
1 1 1 1 1 1 1 1 1 2
```

Output 3

```
2
```

Problem 5

Question

Write a program to accept a non-empty sequence of numbers separated by comma. Print this sequence in the same line separated by comma after removing all duplicate values while preserving the original order. For example:

Input

```
6,5,9,2,6,9,5
```

Output

```
6,5,9,2
```

Answer

```python
# Getting input and after split from blank space assign to l1
l1=input().split(",")
l2=[]
l3=[]
l=len(l1)
# Append each element of l1 in l2 after convert from str to int
for i in l1:
    l2.append(int(i))
# Check each element from l2 if it is not in l3 then append it to l3
for i in l2:
    if i not in l3:
        l3.append(i)
# Print the elements of l3
for i in l3[:-1]:
    print(i,end=",")
print(l3[-1])
```

Test Cases

Public

Input 1

```
6,5,9,2,6,9,5
```

Output 1

```
6,5,9,2
```

Input 2

```
1,2,3,4,5,6,7,8,8,7,6,5,4,3,2,1
```

Output 2

```
1,2,3,4,5,6,7,8
```

Private

Input 1

```
12,24,35,24,88,120,155,88,120,155
```

Output 1

```
12,24,35,88,120,155
```

Input 2

```
1,2,3,4,5,6,7,8,9,0
```

Output 2

```
1,2,3,4,5,6,7,8,9,0
```

Input 3

```
1,1,1,1,1,2,2,2,2,2,2,2,3,3,3,3,3,3,3,4,4,4,5
```

Output 3

```
1,2,3,4,5
```

Input 4

```
-1,-3,-4,-5,1,2,3,4,5
```

Output 4

```
-1,-3,-4,-5,1,2,3,4,5
```

Problem 6

Question

A clockwise rotation of a list consists of taking the last element and moving it to the beginning of the list. For instance, if we rotate the list [1,2,3,4,5], we get [5,1,2,3,4]. If we rotate it again, we get [4,5,1,2,3].

Write a program to accept a non-empty sequence of numbers separated by space and a positive integer `k` and print the list elements in same line separated by space after `k` rotations. For example:

Input

```
1 2 3 4 5
3
```

Output

```
3 4 5 1 2
```

Answer

```python
# Getting input and after split from blank space assign to seq
seq = input().split(' ')
l = []
# Append each element of seq in l after convert from str to int
for i in seq:
    l.append(int(i))
n = len(l)
# Calculate the remainder for reduce the rotation if k is larger than length of l
k = int(input())%n
# Copy all elements from list l to list rt
rt = l[0:]
# Assign number from l at correct place in rt after k rotation
for i in range(0,n):
    rt[i] = l[i - k]
# Print All elements of list rt
for i in range(n-1):
    print(rt[i],end = " ")
print(rt[n-1])
```

Test Cases

Public

Input 1

```
1 2 3 4 5
3
```

Output 1

```
3 4 5 1 2
```

Input 2

```
9 8 7 6 5 4 3 2 1
9
```

Output 2

```
9 8 7 6 5 4 3 2 1
```

Input 3

```
2 3 2 3 2 3 2 3
29
```

Output 3

```
3 2 3 2 3 2 3 2
```

Input 4

Private

Input 1

```
5 4 3 2 1
1
```

Output 1

```
1 5 4 3 2
```

Input 2

```
2 5 6 8 4 9 7 3 1 9 8 6 8
5
```

Output 2

```
1 9 8 6 8 2 5 6 8 4 9 7 3
```

Input 3

```
2 2 2 2 2 2 2 2
5
```

Output 3

```
2 2 2 2 2 2 2 2
```

Input 4

```
3 4 2 1 5 6 2 1 7 8 2 1 9 0 2 1 -2 -5
95
```

Output 4

```
0 2 1 -2 -5 3 4 2 1 5 6 2 1 7 8 2 1 9
```