```
from typing import List, Dict, Set
from doctest import testmod
from collections import Counter
import random
from functools import reduce



# Problem 1

def split_string_into_char_list(s: str)-> List[str]:
    """
    Taking string as a input and returns a list of characters using
the list()

    doctests:
    >>> split_string_into_char_list("epitah")
    ['e', 'p', 'i', 't', 'a', 'h']
    >>> split_string_into_char_list("grudge")
    ['g', 'r', 'u', 'd', 'g', 'e']

    """
    return list(s)




# Problem 2

def form_string_from_char_list(Char_list: List[str])-> str:
    """
    Taking list of strings as input and return a resultant string
using the join()

    Doctest:
    >>> form_string_from_char_list(['e', 'p', 'i', 't', 'a', 'h'])
    'epitah'
    >>> form_string_from_char_list(['g', 'r', 'u', 'd', 'g', 'e'])
    'grudge'
    """

    str1=""
    return (str1.join(Char_list))




# Problem 3

def n_random_number_gen(n: int)-> List[int]:
    """
    Taking input of size n and returns a list filled with random ints
    >>> n_random_number_gen(0)
    []
    """
    return random.sample(range(n), n)
```

```
# Problem 4

def sort_in_desc(li: List[int])-> List[int]:
    """
    Taking  list of int  as input and returns them  in descending
order.

    Doctests:
    >>> sort_in_desc([-5, 100, 71, 0, 78, 798])
    [798, 100, 78, 71, 0, -5]
    >>> sort_in_desc([123, 22, 54, 69])
    [123, 69, 54, 22]
    """
    return sorted(li, reverse=True)




# Problem 5
def freq(li: List[int]) -> Dict[int, int]:
    """
    Taking list as input and Counts the frequency of elements present
in the list and return a dictionary with key as elements

    Doctests:
    >>> freq([-1,-1,-1,-1,1,1,2,2,2,2,2,2,0,0,0,3,3,3,3,3,3,3,3])
    {-1: 4, 1: 2, 2: 6, 0: 3, 3: 8}
    >>> freq([0,2,3,0,2,3,0,2,3,3])
    {0: 3, 2: 3, 3: 4}
    """

    d ={}
    for ele in li:
        if ele in d:
            d[ele]=d[ele]+1
        else:
            d[ele]=1
    return d




# Problem 6

def unique_ele_list(li: List[int]) -> Set[int]:
    """
    Taking  list of int as input and returns only unique elements
present in list

    Doctests:
    >>> unique_ele_list([0,0,0,2,3,4,5])
    {0, 2, 3, 4, 5}
    >>> unique_ele_list([6, 7, 8, 9, 1, 6, 7 ,7 ,8 ,0])
    {0, 1, 6, 7, 8, 9}
    """
    s=set()
    for ele in li:
        s.add(ele)
    return s



# Problem 7
def first_repeating_ele(li: List):
    """
    Taking a list of int as input and returns first element that is 
repeated

    >>> first_repeating_ele([1, 0, 3, 4, 2, 1])
    1
    >>> first_repeating_ele(["x", "b", "b"])
    'b'
    """
    s=set()
    for ele in li:
        if(ele in s):
            return ele
        s.add(ele)
    return None



# Problem 8
def n_to_dict(n: int) -> Dict[int, List[int]]:
    """
    Taking n as input and returns a dictionary with  list of squares 
and cubes

    Doctests:
    >>> n_to_dict(4)
    {0: [0, 0], 1: [1, 1], 2: [4, 8], 3: [9, 27], 4: [16, 64]}
    >>> n_to_dict(0)
    {0: [0, 0]}
    >>> n_to_dict(-2)
    {}
    """

    d={}
    for i in range(0,n+1):
        li=[i**2,i**3]
        d[i]=li
    return d



# Problem 9
def zip_elements(li1: List, li2: List) -> List:
    """
    Takes two lists of same length and returns zipping them together 
as one new list
```

```
    Doctests:
    >>> zip_elements([-1, -2, -3], [4, 5, 6])
    [(-1, 4), (-2, 5), (-3, 6)]
    >>> zip_elements(["hell"], ["word"])
    [('hell', 'word')]
    """
    result = list(zip(li1 , li2))
    return result




# Problem 10

def gen_squares(n: int) -> List[int]:
    """
    Taking a int n as input and returns the list of squares ranging 
from 0, 1, 2, ..., n

    Doctests:
    >>> gen_squares(-3)
    []
    >>> gen_squares(5)
    [0, 1, 4, 9, 16, 25]
    """
    li=[]
    for i in range(0,n+1):
        li.append(i)

    li_new = [ele**2 for ele in li]
    return li_new




# Problem 11
def gen_square_dict(n: int) -> Dict[int, int]:
     """
    Taking n as input and returns squares from 0, 1, 2, ..., n

    >>> gen_square_dict(-1)
    {}
    >>> gen_square_dict(5)
    {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
    """
     return {i:i*i for i in range(n+1)}




# Problem 12
def Squares(x: int) -> int:
    """
    Taking x as input and Return square of n
    """
    return x * x
```

```python
class MyClass:

    def __init__(self, li: List[int]) -> None:
        """Saves a copy of nums in a local variable"""
        self.li = li[:]

    def apply(self, func_name) -> List:
        """
        Applies the function to the list and returns the values

        Doctests:
        >>> l = MyClass([1,2,3,4,5])
        >>> print(l.apply(Squares))
        [1, 4, 9, 16, 25]
        >>> print(l.apply(lambda y: y * y))
        [1, 4, 9, 16, 25]
        >>> a = MyClass(["a", "b"])
        >>> a.apply(Squares)
        given function can't execute the class list
        """
        try:
            return list(map(func_name, self.li))
        except TypeError:
            print("given function can't execute the class list")




# Problem 13
def low_to_upper(string: List[str]) -> List[str]:
    """
    Taking list of strings as input and return in all strings in
capital letter

    >>> low_to_upper(["hell", "word"])
    ['HELL', 'WORD']
    >>> low_to_upper([])
    []
    """
    return list(map(str.upper, string))



# Problem 14
def product(x,y):
    return (x*y)

def mult(nums: List[int]) -> List[int]:
    """
    Taking  list of int as input and return product of all  elements
in list

    >>> mult([1, 2, 3, 4, 5, 6])
    720
    >>> mult([0, 23, 4, 2, 9, -1])
    0
    """
    return reduce(product, nums)



if __name__ == "__main__":
    testmod()
```