LAB 04

Dictionaries, Sets and Lists

1. Write four functions that directly mutate a list:

- repeat(lst, n): Repeat lst n times.
- add(lst, x): Adds x to the end of the lst.
- remove(lst, m, n): Removes all elements between indices m and n inclusive in lst.
- concat(lst, x): concatenates lst with x (another list).

Examples:

lst = [1, 2, 3, 4]
repeat(lst, 3) ➞ [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
add(lst, 1) ➞ [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1]
remove(lst, 1, 12) ➞ [1]
concat(lst, [3, 4]) ➞ [1, 3, 4]

2. Create a function that takes a list lst and a number N and returns a list of two integers from lst whose product equals N.

Examples:

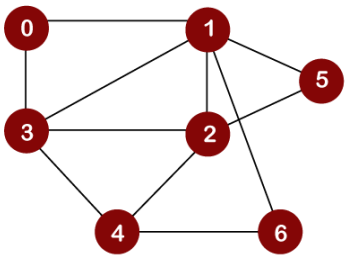
two_product([1, 2, -1, 4, 5], 20) ➞ [4, 5]
two_product([1, 2, 3, 4, 5], 10) ➞ [2, 5]
two_product([100, 12, 4, 1, 2], 15) ➞ None

3. Create a function that takes a list lst and a number N and returns a list of two integers from lst whose product equals N.

Examples:

two_product([1, 2, -1, 4, 5], 20) ➞ [4, 5]
two_product([1, 2, 3, 4, 5], 10) ➞ [2, 5]
two_product([100, 12, 4, 1, 2], 15) ➞ None

4. Create an dictionary representation of the following graph and run the following BFS code




```
        graph = {
  '5' : ['3','7'],
  '3' : ['2', '4'],
  '7' : ['8'],
  '2' : [],
  '4' : ['8'],
  '8' : []
}

visited = [] # List for visited nodes.
queue = []     #Initialize a queue
```

```
def bfs(visited, graph, node): #function for BFS
  visited.append(node)
  queue.append(node)

  while queue:          # Creating loop to visit each node
    m = queue.pop(0)
    print (m, end = " ")

    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

# Driver Code
print("Following is the Breadth-First Search")
bfs(visited, graph, '5')    # function calling
```

5. Write following functions using sets:

a. Unioin(s1,s2)
b. Intersection(s1,s2)