In [31]:
```python
def invertdict(d):
    return {v: k for k, v in d.items()}

print(invertdict({"a": 1, "b": 2, "c": 3}))
# {1: 'a', 2: 'b', 3: 'c'}
```

```
{1: 'a', 2: 'b', 3: 'c'}
```

In [32]:
```python
def valuesort(d):
    return [d[k] for k in sorted(d.keys())]

print(valuesort({"b": 3, "a": 1, "c": 2}))
# [1, 3, 2]
```

```
[1, 3, 2]
```

In [33]:
```python
def n_largest(lst, n):
    return sorted(lst, reverse=True)[:n]

print(n_largest([4, 1, 9, 12, 5, 6], 3))
# [12, 9, 6]
```

```
[12, 9, 6]
```

In [34]:
```python
from itertools import product

string_maps = {
    "1": "abc", "2": "def", "3": "ghi", "4": "jkl",
    "5": "mno", "6": "pqrs", "7": "tuv",
    "8": "wxy", "9": "z"
}

def digit_combinations(digits):
    groups = [string_maps[d] for d in digits if d in string_maps]
    return [''.join(p) for p in product(*groups)]

print(digit_combinations("23"))
# ['dg', 'dh', 'di', 'eg', 'eh', 'ei', 'fg', 'fh', 'fi']
```

```
['dg', 'dh', 'di', 'eg', 'eh', 'ei', 'fg', 'fh', 'fi']
```

In [35]:
```python
def num_islands(grid):
    n, m = len(grid), len(grid[0])
    visited = [[False]*m for _ in range(n)]

    def dfs(r, c):
        if r < 0 or c < 0 or r >= n or c >= m or grid[r][c] == '0' or visited[r]
            return
        visited[r][c] = True
        dfs(r+1, c); dfs(r-1, c); dfs(r, c+1); dfs(r, c-1)

    count = 0
    for i in range(n):
        for j in range(m):
            if grid[i][j] == '1' and not visited[i][j]:
                dfs(i, j)
                count += 1
    return count

data = [
```

```
"1100000111","1000000111","0000000111","0010001000","0000011100",
"0000111110","0001111111","1000111110","1100011100","1110001000"
]

print("Number of islands:", num_islands(data))
# 5
```

```
Number of islands: 5
```

In [36]:
```
def double_numbers(s):
    nums = [int(x.strip()) for x in s.split(",")]
    return [2*x for x in nums]

print(double_numbers("123, 456, 222, 145"))
# [246, 912, 444, 290]

print(double_numbers("-1, 0, -2, 2, 0, 1"))
# [-2, 0, -4, 4, 0, 2]
```

```
[246, 912, 444, 290]
[-2, 0, -4, 4, 0, 2]
```

In [37]:
```
import glob

def extract_chars(folder="*.txt"):
    chars = []
    for file in glob.glob(folder):
        with open(file, "r") as f:
            chars.extend(list(f.read()))
    return chars

print(extract_chars("*.txt"))
```

```
['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '\n', 'P', 'y', 't', 'h',
'o', 'n', ' ', 'P', 'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g', '\n', 'h',
'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', ' ', 'h', 'e', 'l', 'l', 'o', '
', 'p', 'y', 't', 'h', 'o', 'n', ' ', 'p', 'y', 't', 'h', 'o', 'n', ' ', 'p',
'y', 't', 'h', 'o', 'n', ' ', 'c', 'o', 'd', 'i', 'n', 'g', ' ', 'i', 's', ' ',
'f', 'u', 'n', ' ', 'h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', '\n',
'P', 'o', 't', 'a', 't', 'o', ' ', 'C', 'h', 'i', 'p', 's', '|', '2', '0', '\n',
'P', 'o', 'p', 'c', 'o', 'r', 'n', '|', '3', '0', '\n', 'C', 'h', 'o', 'c', 'o',
'l', 'a', 't', 'e', '|', '1', '5', '\n', 'B', 'i', 's', 'c', 'u', 'i', 't', '|',
'1', '0', '\n', 'S', 'o', 'f', 't', ' ', 'D', 'r', 'i', 'n', 'k', '|', '1', '2',
'\n']
```

In [38]:
```
def vending_machine():
    items = {}
    with open("VendingItems.txt") as f:
        for line in f:
            name, price = line.strip().split("|")
            items[name] = int(price)

    while True:
        item = input("Enter item: ")
        if item not in items:
            print("Available Items are", list(items.keys()))
            continue
        break

    while True:
        try:
```

```
            money = int(input("Enter money: "))
            break
        except ValueError:
            print("Bad Input. Try Again.")

    price = items[item]
    if money < price:
        print("Insufficient money! Need", price - money, "more.")
    else:
        print("Thank you for your purchase. Enjoy")
        if money > price:
            print("Do not forget to collect your change,", money - price, "Rs.")

vending_machine()
```

```
Thank you for your purchase. Enjoy
Do not forget to collect your change, 5 Rs.
```

In [39]:
```
def binary_search_recursive(arr, target, low, high):
    if low > high:
        return -1
    mid = (low + high)//2
    if arr[mid] == target: return mid
    elif arr[mid] > target: return binary_search_recursive(arr, target, low, mid
    else: return binary_search_recursive(arr, target, mid+1, high)

def binary_search_iterative(arr, target):
    low, high = 0, len(arr)-1
    while low <= high:
        mid = (low + high)//2
        if arr[mid] == target: return mid
        elif arr[mid] < target: low = mid+1
        else: high = mid-1
    return -1

arr = [1,3,5,7,9,11]
print(binary_search_recursive(arr, 7, 0, len(arr)-1))
print(binary_search_iterative(arr, 7))
```

```
3
3
```

In [40]:
```
from collections import Counter

def most_repetitive_word(filename):
    with open(filename) as f:
        words = f.read().split()
    return Counter(words).most_common(1)[0]

print(most_repetitive_word("sample.txt"))
```

```
('hello', 3)
```

In [41]:
```
def unique_list(lst):
    return list(set(lst))

print(unique_list([1,2,2,3,4,4,5]))
# [1, 2, 3, 4, 5] (order may vary)
```

```
[1, 2, 3, 4, 5]
```

In [42]:
```
from collections import Counter

def first_non_repeating(s):
    count = Counter(s)
    for ch in s:
        if count[ch] == 1:
            return ch
    return None

print(first_non_repeating("aabbcdeff"))
# c
```

```
c
```

In [43]:
```
import re

def remove_parenthesis(lst):
    return [re.sub(r"\s*\(.*?\)", "", s).strip() for s in lst]

data = ["example (.com)", "w3resource", "github (.com)", "stackoverflow (.com)"]
print(remove_parenthesis(data))
# ['example', 'w3resource', 'github', 'stackoverflow']
```

```
['example', 'w3resource', 'github', 'stackoverflow']
```

In [ ]:
```

```