Week 10 q1

Given list=Input: strs = ["eat", "tea", "tan", "ate", "nat", "bat"]

Histogram for “eat”

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Key=10001000000000000001000000

Value=”ear”

Histogram for “tea”

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Key=10001000000000000001000000

Value=”tea”

Histogram for “tan”

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0

Key=10000000000001000001000000

Value=”tan”

Histogram for “ate”

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Key=10001000000000000001000000

Value=”ate”

Histogram for “nat”

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0

Key=10000000000001000001000000

Value=”nat”

Histogram for “bat”

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Key=11000000000000000001000000

Value=”bat”

Now grouping all the words with same keys

Final answer is

Output= Output: [["bat"],["nat","tan"],["ate","eat","tea"]]

Chat gpt code

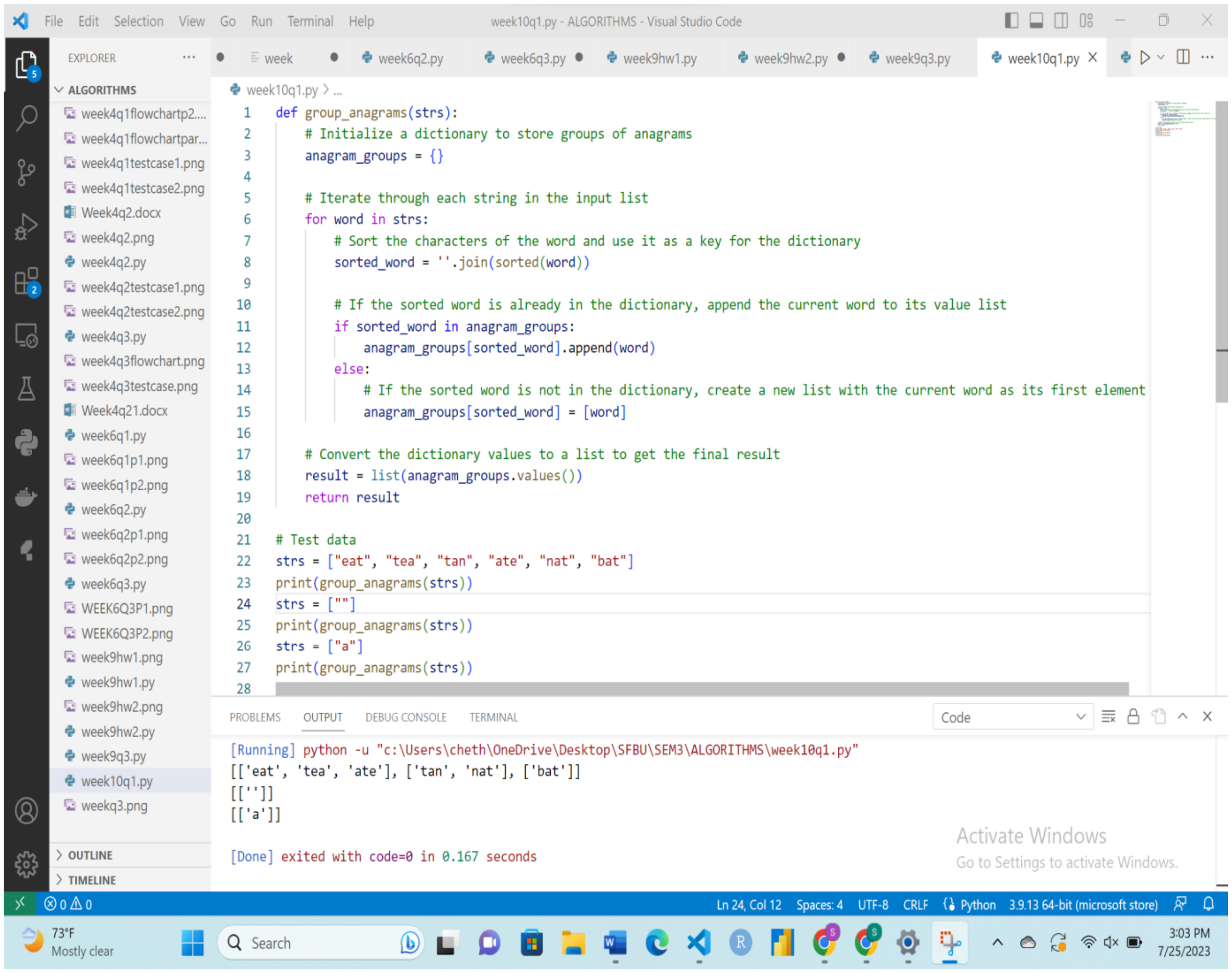
```
def group_anagrams(strs):
    # Initialize a dictionary to store groups of anagrams
    anagram_groups = {}

    # Iterate through each string in the input list
    for word in strs:
        # Sort the characters of the word and use it as a key for the dictionary
        sorted_word = ''.join(sorted(word))
        # If the sorted word is already in the dictionary, append the current
word to its value list
        if sorted_word in anagram_groups:
            anagram_groups[sorted_word].append(word)
        else:
            # If the sorted word is not in the dictionary, create a new list with
the current word as its first element
            anagram_groups[sorted_word] = [word]

    # Convert the dictionary values to a list to get the final result
    result = list(anagram_groups.values())
    return result

# Test data
strs = ["eat", "tea", "tan", "ate", "nat", "bat"]
print(group_anagrams(strs))
```

Test case



```python
def group_anagrams(strs):
    # Initialize a dictionary to store groups of anagrams
    anagram_groups = {}

    # Iterate through each string in the input list
    for word in strs:
        # Sort the characters of the word and use it as a key for the dictionary
        sorted_word = ''.join(sorted(word))

        # If the sorted word is already in the dictionary, append the current word to its value list
        if sorted_word in anagram_groups:
            anagram_groups[sorted_word].append(word)
        else:
            # If the sorted word is not in the dictionary, create a new list with the current word as its first element
            anagram_groups[sorted_word] = [word]

    # Convert the dictionary values to a list to get the final result
    result = list(anagram_groups.values())
    return result

# Test data
strs = ["eat", "tea", "tan", "ate", "nat", "bat"]
print(group_anagrams(strs))
strs = [""]
print(group_anagrams(strs))
strs = ["a"]
print(group_anagrams(strs))
```

```
[Running] python -u "c:\Users\cheth\OneDrive\Desktop\SFBU\SEM3\ALGORITHMS\week10q1.py"
[['eat', 'tea', 'ate'], ['tan', 'nat'], ['bat']]
[['']]
[['a']]

[Done] exited with code=0 in 0.167 seconds
```