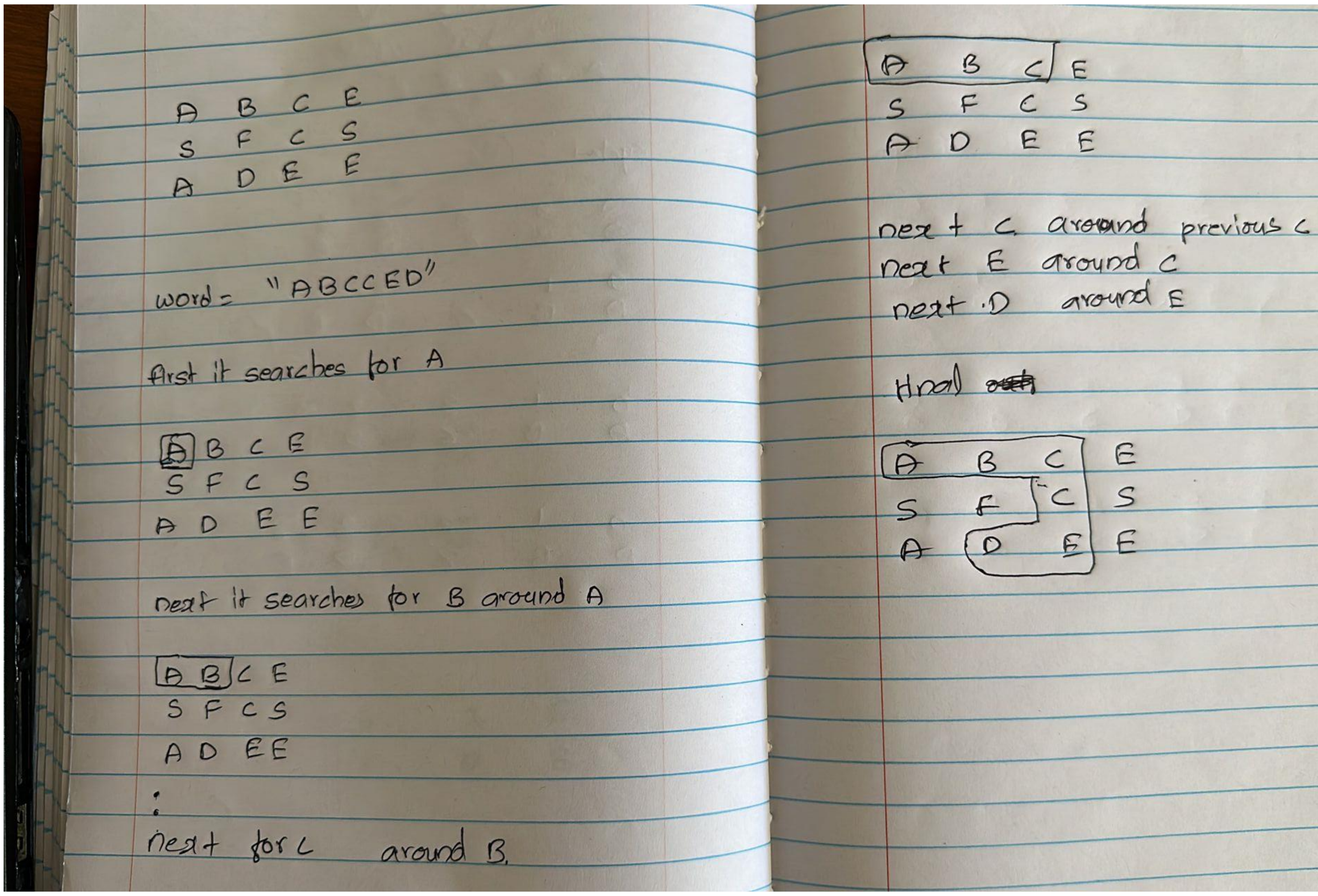
Week6q3



A B C E
S F C S
A D E E

word = "ABCCED"

first it searches for A

A B C E
S F C S
A D E E

next it searches for B around A

A B C E
S F C S
A D E E

:

next for C around B.

A B C E
S F C S
A D E E

next C around previous C
next E around C
next D around E

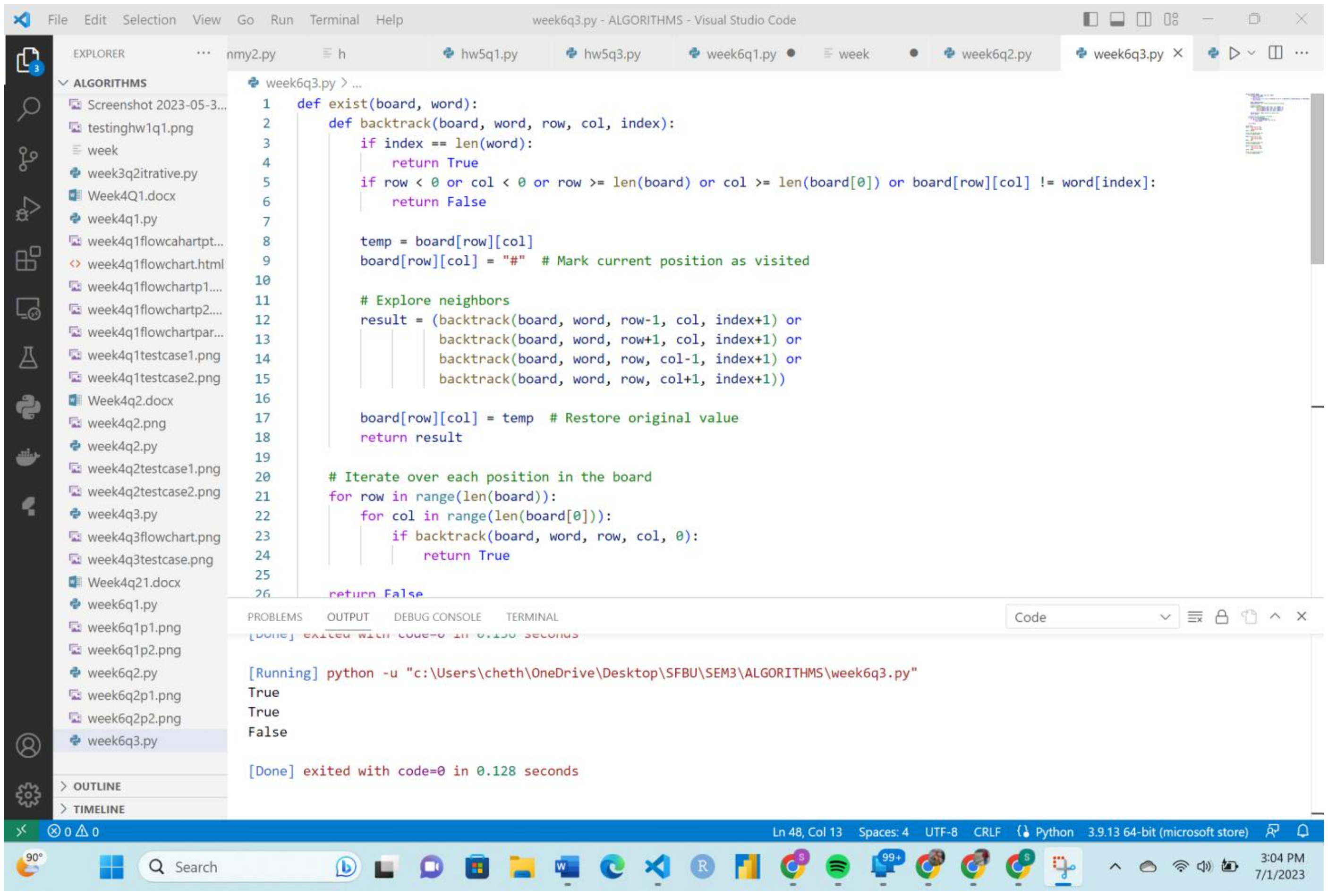
final ~~[illegible]~~

A B C E
S F C S
A D E E

Chatgpt code

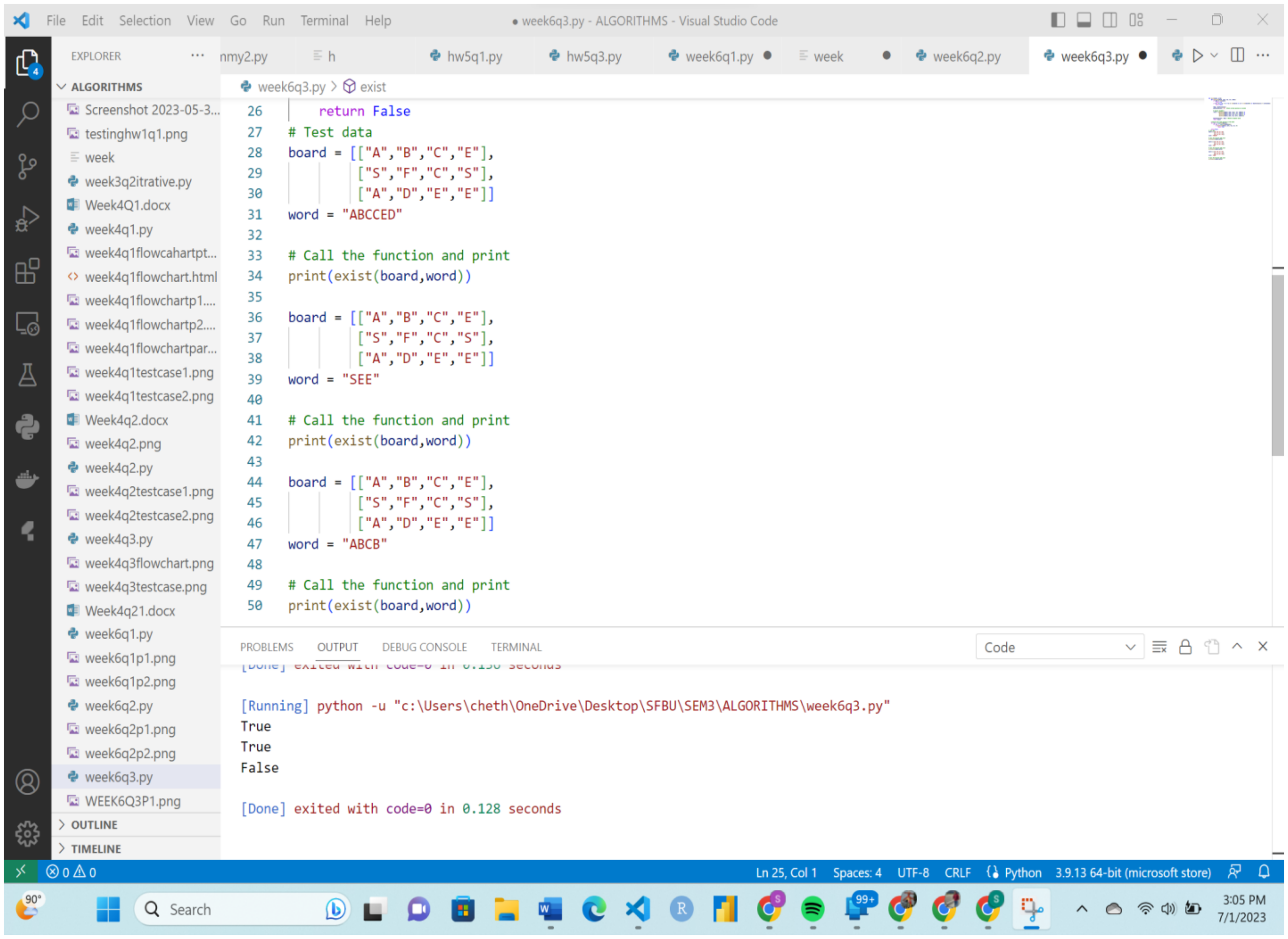
```
def exist(board, word):
    def backtrack(board, word, row, col, index):
        if index == len(word):
            return True
        if row < 0 or col < 0 or row >= len(board) or col >= len(board[0]) or 
board[row][col] != word[index]:
            return False

        temp = board[row][col]
        board[row][col] = "#"  # Mark current position as visited

        # Explore neighbors
        result = (backtrack(board, word, row-1, col, index+1) or
                  backtrack(board, word, row+1, col, index+1) or
                  backtrack(board, word, row, col-1, index+1) or
                  backtrack(board, word, row, col+1, index+1))

        board[row][col] = temp  # Restore original value
        return result

    # Iterate over each position in the board
    for row in range(len(board)):
        for col in range(len(board[0])):
            if backtrack(board, word, row, col, 0):
                return True

    return False

# Test data
board = [["A","B","C","E"],
         ["S","F","C","S"],
         ["A","D","E","E"]]
word = "ABCCED"

# Call the function and print
print(exist(board,word))
```

test case



```
def exist(board, word):
    def backtrack(board, word, row, col, index):
        if index == len(word):
            return True
        if row < 0 or col < 0 or row >= len(board) or col >= len(board[0]) or board[row][col] != word[index]:
            return False

        temp = board[row][col]
        board[row][col] = "#"  # Mark current position as visited

        # Explore neighbors
        result = (backtrack(board, word, row-1, col, index+1) or
                  backtrack(board, word, row+1, col, index+1) or
                  backtrack(board, word, row, col-1, index+1) or
                  backtrack(board, word, row, col+1, index+1))

        board[row][col] = temp  # Restore original value
        return result

    # Iterate over each position in the board
    for row in range(len(board)):
        for col in range(len(board[0])):
            if backtrack(board, word, row, col, 0):
                return True

    return False
```

```
[Running] python -u "c:\Users\cheth\OneDrive\Desktop\SFBU\SEM3\ALGORITHMS\week6q3.py"
True
True
False

[Done] exited with code=0 in 0.128 seconds
```



```
    return False
# Test data
board = [["A","B","C","E"],
        ["S","F","C","S"],
        ["A","D","E","E"]]
word = "ABCCED"

# Call the function and print
print(exist(board,word))

board = [["A","B","C","E"],
        ["S","F","C","S"],
        ["A","D","E","E"]]
word = "SEE"

# Call the function and print
print(exist(board,word))

board = [["A","B","C","E"],
        ["S","F","C","S"],
        ["A","D","E","E"]]
word = "ABCB"

# Call the function and print
print(exist(board,word))
```

```
[Running] python -u "c:\Users\cheth\OneDrive\Desktop\SFBU\SEM3\ALGORITHMS\week6q3.py"
True
True
False

[Done] exited with code=0 in 0.128 seconds
```