Name :M.vijay kumar

Htno :2403A51445

Batch :17

**M . 1:**

**Prompt :**

Write a Python program that reads employee data from a CSV (with columns name,dept,salary), and outputs a new CSV where rows are sorted by dept in ascending order and by salary in descending order within each department.

Input as :

name,dept,salary  
Raj,Eng,120  
Maya,HR,90  
Abi,Eng,110

**code :**

import csv

def stable\_sort\_employees(input\_file, output\_file):

    with open(input\_file, newline='') as infile:

        reader = csv.DictReader(infile)

        rows = list(reader)

    # Sort: dept ascending, salary descending

    rows.sort(key=lambda r: (r['dept'], -int(r['salary'])))

    # Write back to CSV

    with open(output\_file, 'w', newline='') as outfile:

        writer = csv.DictWriter(outfile, fieldnames=['name', 'dept', 'salary'])

        writer.writeheader()

        writer.writerows(rows)

# ---- Example Run ----

# Create sample.csv first for demonstration

with open("sample.csv", "w", newline="") as f:

    f.write("name,dept,salary\nRaj,Eng,120\nMaya,HR,90\nAbi,Eng,110\n")

stable\_sort\_employees("sample.csv", "sorted.csv")

with open("sorted.csv") as f:

    print(f.read())

**output :**

**A screenshot of a computer

AI-generated content may be incorrect.**

**M.2 :**

**Prompt :**

Write a Python function process\_movements(commands) that takes a list of strings representing movement commands on a 2D grid and returns the final (x, y) position starting from (0, 0).

If a command is invalid , ignore it.

Update coordinates

N: y += steps,S: y -= steps,E: x += steps,W: x -= steps

**Code :**

def process\_movements(commands):

    x, y = 0, 0  # Start at origin

    for cmd in commands:

        if len(cmd) < 2:

            continue  # ignore invalid token like 'N' or ''

        direction, steps\_str = cmd[0], cmd[1:]

        if not steps\_str.isdigit():

            continue  # ignore if steps are not numbers

        steps = int(steps\_str)

        if direction == 'N':

            y += steps

        elif direction == 'S':

            y -= steps

        elif direction == 'E':

            x += steps

        elif direction == 'W':

            x -= steps

        else:

            continue  # ignore invalid direction

    return (x, y)

# ---- Example ----

sample\_input = ['N2', 'E1', 'S1', 'E2']

print(process\_movements(sample\_input))  # ✅ (3, 1)

**output :**

