NAME:GUGGILLA ANUJA

HALLTICKET NO:2403A51101

BATCH:06

SET:SUBGROUP—M

Task:M.1

PROMPT:

HR requires deterministic sorting of employee data for payroll audits.
Sort employees from a CSV file by:Department (dept) in ascending order.Salary (salary) in descending order.Sorting must be stable (preserve original order for equal values).
Use csv.DictReader and csv.DictWriter to read and write the CSV.

CODE:

LAB.M.1.py > ...

```python
import csv
from typing import List

Zencoder
def sort_employees(input_csv: str, output_csv: str) -> None:
    with open(input_csv, newline='') as infile:
        reader = list(csv.DictReader(infile))
        # Stable sort: dept ascending, salary descending
        sorted_rows = sorted(reader, key=lambda x: (x['dept'], -int(x['salary'])))
    with open(output_csv, 'w', newline='') as outfile:
        writer = csv.DictWriter(outfile, fieldnames=reader[0].keys())
        writer.writeheader()
        writer.writerows(sorted_rows)

# Example usage and output
if __name__ == "__main__":
    # Sample data for demonstration
    employees = [
        {'name': 'Alice', 'dept': 'HR', 'salary': '5000'},
        {'name': 'Bob', 'dept': 'IT', 'salary': '7000'},
        {'name': 'Charlie', 'dept': 'HR', 'salary': '6000'},
        {'name': 'David', 'dept': 'IT', 'salary': '6500'},
        {'name': 'Eve', 'dept': 'Finance', 'salary': '5500'}
    ]
    # Write sample data to input.csv
    with open('input.csv', 'w', newline='') as f:
        writer = csv.DictWriter(f, fieldnames=['name', 'dept', 'salary'])
```

```python
# Example usage and output
if __name__ == "__main__":
    # Sample data for demonstration
    employees = [
        {'name': 'Alice', 'dept': 'HR', 'salary': '5000'},
        {'name': 'Bob', 'dept': 'IT', 'salary': '7000'},
        {'name': 'Charlie', 'dept': 'HR', 'salary': '6000'},
        {'name': 'David', 'dept': 'IT', 'salary': '6500'},
        {'name': 'Eve', 'dept': 'Finance', 'salary': '5500'}
    ]
    # Write sample data to input.csv
    with open('input.csv', 'w', newline='') as f:
        writer = csv.DictWriter(f, fieldnames=['name', 'dept', 'salary'])
        writer.writeheader()
        writer.writerows(employees)
    # Sort and output to output.csv
    sort_employees('input.csv', 'output.csv')
    # Print output.csv contents
    with open('output.csv', newline='') as f:
        print(f.read())
```

OUTPUT:

```
PS C:\Users\Administrator\OneDrive\ai> & C:/Python313/python.exe c:/Users/Administrator/OneDrive/ai/LAB.M.1.py
name,dept,salary
Eve,Finance,5500
Charlie,HR,6000
Alice,HR,5000
Bob,IT,7000
David,IT,6500

PS C:\Users\Administrator\OneDrive\ai>
```

OBSERVATION: The program reads the CSV correctly using DictReader.
Sorting is stable using dept ascending and salary descending.
The output CSV preserves headers and correct row order.

Task:M.2

PROMPT:

Move an agent on a grid from (0,0) using commands like N2, E1, S3, W4.

- N increases y, E increases x, S decreases y, W decreases x.
- Ignore invalid commands (wrong direction or non-numeric steps).
  Return the final (x, y) after applying valid commands

CODE:

```python
def process_movements(commands):
    x, y = 0, 0
    valid_directions = {'N', 'E', 'S', 'W'}
    for cmd in commands:
        if len(cmd) < 2:
            continue  # Invalid if too short
        direction = cmd[0]
        steps = cmd[1:]

        if direction not in valid_directions:
            continue  # Invalid direction

        if not steps.isdigit():
            continue  # Steps must be numeric
        steps = int(steps)
        if direction == 'N':
            y += steps
        elif direction == 'S':
            y -= steps
        elif direction == 'E':
            x += steps
        elif direction == 'W':
            x -= steps

    return (x, y)
if __name__ == "__main__":
    commands1 = ['N2', 'E1', 'S1', 'E2']
    print("Output:", process_movements(commands1))  # Expected (3
    commands2 = ['N3', 'X5', 'E2', 'W1', 'S-1', 'Eabc']
    print("Output with invalids:", process_movements(commands2))
```

OUTPUT:

```
[Running] python -u "c:\Users\admin\OneDrive\Desktop\BTECH IIYR\AI\LABEXAM2.
py"
Output: (3, 1)
Output with invalids: (1, 3)

[Done] exited with code=0 in 0.073 seconds
```

OBSERVATION: The program starts at (0,0) and updates position based on valid commands.

Invalid directions and non-numeric steps are ignored.

The final position is calculated correctly, matching the expected output.