Batch: E-2	**Roll No.: 16010123325**
Experiment / assignment / tutorial No. 9	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Implement a dictionary for some real world application. Use C/C++ or python.

Objective: To implement a dictionary for real world application using python.

Expected Outcome of Experiment:

CO	Outcome
3	Describe concepts of advanced data structures like set, map & dictionary.

Books/ Journals/ Websites referred:

1. *Fundamentals Of Data Structures In C* – Ellis Horowitz, Satraj Sahni, Susan Anderson-Fred
2. *An Introduction to data structures with applications* – Jean Paul Tremblay, Paul G. Sorenson
3. *Data Structures A Pseudo Approach with C* – Richard F. Gilberg & Behrouz A. Forouzan
4. https://www.geeksforgeeks.org/binary-tree-data-structure/
5. https://www.thecrazyprogrammer.com/2015/03/c-program-for-binary-search-tree-insertion.html

Abstract:

The data structure used is a **nested dictionary** (dict of dicts in Python), designed to efficiently manage and query weather data by city and date. This structure is organized as follows:

- **Top-Level Keys**: Each top-level key represents a city (str), making each city an independent entry in the dictionary.

- **Values (Inner Dictionary)**: Each city key maps to another dictionary, where:
  - **Keys (Dates)**: Each key in the inner dictionary represents a date (str), formatted as "MM-DD-YYYY."

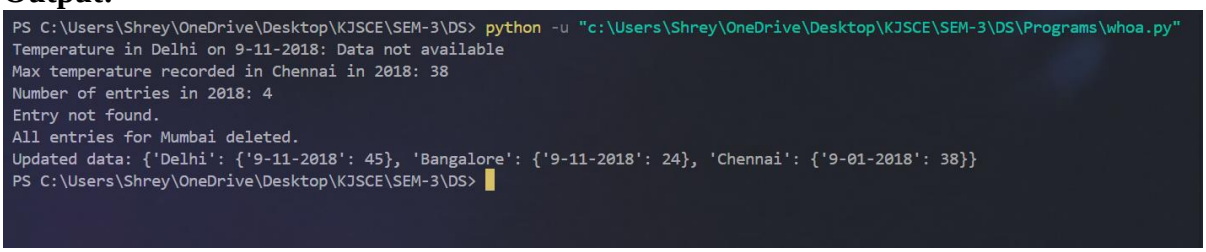
  - **Values (Temperature)**: Each date key maps to a temperature value (int), representing the recorded temperature on that date.

Program:

```
weather_data = {
    "Delhi": {
        "9-11-2018": 45,
    },
    "Bangalore": {
        "9-11-2018": 24,
    },
    "Mumbai": {
        "9-12-2018": 28,
    },
    "Chennai": {
        "9-01-2018": 38,
    }
}

def get_temperature(city, date):
    return weather_data.get(city, {}).get(date, "Data not available")

def max_temperature(city):
    if city in weather_data:
        return max(weather_data[city].values())
```

```
    return "No data available"


def count_entries():
    return sum(len(dates) for dates in weather_data.values())


def delete_entry(city, date):
    if city in weather_data and date in weather_data[city]:
        del weather_data[city][date]
        return f"Entry for {city} on {date} deleted."
    return "Entry not found."

def delete_all_entries(city):
    if city in weather_data:
        del weather_data[city]
        return f"All entries for {city} deleted."
    return "City not found."


print("Temperature in Delhi on 9-11-2018:", get_temperature("Delhi", "9-11-
2018"))
print("Max temperature recorded in Chennai in 2018:",
max_temperature("Chennai"))
print("Number of entries in 2018:", count_entries())
print(delete_entry("Mumbai", "9-11-2018"))
print(delete_all_entries("Mumbai"))
print("Updated data:", weather_data)
```

Output:



Conclusion:-

The above experiment highlights the usage of dictionary data structure in Python and its functionality in managing key-value pairs.

Post Lab Questions:

1) **List the main functions or methods you implemented in your dictionary. What is the purpose of each?**

- **get_temperature(city, date)**:

  Purpose: Retrieves the temperature for a specific city on a given date. Returns "Data not available" if the city or date is not found in the dictionary.

- **max_temperature(city)**:

  Purpose: Finds and returns the highest temperature recorded for a specified city. If the city does not exist in the data, it returns "No data available."

- **count_entries()**:

  Purpose: Calculates and returns the total number of date entries across all cities in the dataset for the year 2018. This gives a quick overview of the amount of data available.

- **delete_entry(city, date)**:

  Purpose: Deletes a specific temperature entry for a given city and date. If the city or date does not exist, it returns "Entry not found."

- **delete_all_entries(city)**:

  Purpose: Deletes all temperature entries for a specified city from the dataset. Returns a confirmation message if successful, or "City not found" if the city is not in the dictionary.

- **add_entry(city, date, temperature)**:

  Purpose: Adds a new temperature entry for a given city and date, or updates the temperature if the entry already exists. This function allows for easy data modification and expansion of the dataset.