Task 4:- Implement various Searching and Sorting operations in python programing

Aim:

To Implement various searching and sorting operations in python programming.

4.1

1. A company stores employee records in a list dictionaries, where each dictionary contains id, name and department. Write a function find_employee_by_id that takes this list and a target employee ID as arguments and returns the dictionary of the employee with the matching ID, or None if no such employee is found.

Algorithm:

1. Input Definition:
2. Define the function find_employee_by_id that takes two parameters:
   a. A list of dictionary (employees), where each dictionary represents an employee record with key id, name, and department.
   b. An integer (target_id) representing the employee ID to be searched.
3. Iterate Trough the list: Use a for loop to iterate through each dictionary in the employees list.
4. Check for Matching ID: Within the loop, check if the id field of the current dictionary matches the target_id.
5. Return Macthing Record: If a match is found, return the current dictionary.
6. Handle No match: If the loop completes without finding a match return None.

Output:

{'id': 2, 'name': 'Bob', 'department': 'Engineering'}

Program:

```
def find_employee_by_id (employees, target_id):
 for employee in employees:
    if employee['id'] == target_id:
       return employee
 return None
# Test the function
employee = [
 {'id': 'name': 'Alice', ' department': 'HR'},
 {'ide': 2,'name':'Bob', 'department': Engineering'};
 {'id': 3,'name': 'Charlie', 'department': 'Sales'},
]
print (find_employee_by_id(employee, 2)) # output:
{'id': 2','name':'Bob', 'department': 'Engineering'}
```

4.2: You are developing a grade management system for a school. The system maintains a list of student records, where each record is represented as a dictionary containing a student's name and score. Your task is to implement a feature that sort the student record by their scores using the Bubble sort algorithm.

Algorithm:

1. Initialization:
   - Get the length of the students like and store it in n.
2. Outer Loop.
   - Iterate from i=0 to n-1 (inclusive). This loop represents the number of passes through the list
3. Track Swaps:
   - Initialize a boolean variable swapped to false. This variable will track if any swaps are made in the current passes.
4. Inner loop:
   - Iterate from j=0 to n-i-2 (inclusive). This loop compares adjacent elements in the list and performs swaps if necessary.
5. Compare and swap:
   - For each pair of adjacent element (i.e, student[j] and students[j+1]):
     - Compare thoirs score values
     - If students[j]['score'] > students[j+1]['score'], swap the two elements.
     - Set swapped to True to indicate that a swap was made.

Output:-

Before sorting:

{'name': 'Alice', 'score': 88}

{'name': 'Bob', 'score': 95}

{'name': 'Charlie', 'score': 75}

{'name': 'Diana', 'score': 85}

After sorting:

{'name': 'charlie', 'sorce': 75}

{'name': 'Diana', 'score': 85}

{'name': 'Alice', 'score': 88}

{'name': 'Bob', 'score': 95}

6. Early Termination:

- After each pass of the inner loop, check if swapped is false. If no swaps were made during the pass, the list is already sorted, and you can break out of the outer loop early.

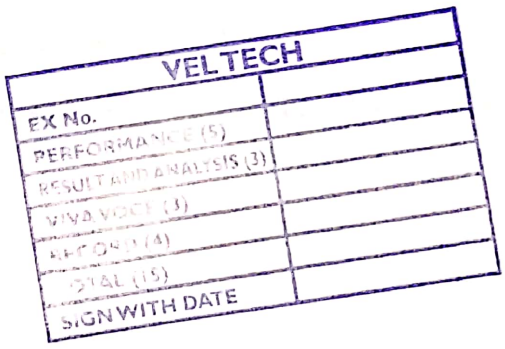
7. Completion:

- The function modifies the students list in place, sorting it by score.

Program:

```
def bubble_sort_score (student):
    n = len (students)
    for i in range (n):
    # Track if any swap is made in this pass
    swapped = False
    for j in range (0, n-i-1):
        if students [j]['score'] > students [j+1]['score']:
        # Swap if the score of the current student
            is greater than the next.
        students [j], student[j+1] = students [j+1], student [j]
        swapped = True
    # If no two elements were swapped, the list is
     already sorted.
     if not swapped:
      break:
# Example usage
Student = [
  {'name': 'Alice', 'score': 88},
  {'name': 'Bob', 'score': 95},
  {'name': 'Charlie', 'score': 75},
  {'name': 'Diana', 'score': 85}
]
```

```
print("Before sorting:")
for student in a students:
    print(student)
bubble_sort_score(students)
print("\nAfter sorting:")
for student in students:
    print(student)
```



Result:-

Thus, the program for various Searching and sorting operation is executed and verified successfully