5.1 A Company stores employee records as a list of dictionaries. Each dictionary contains id, name, and department. Write a function to 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 dictionaries (employees_by_id that takes two parameters)

3. Iterate Through 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 Matching Record:

If a match is found, return the current dictionary

6. Handle No Match:

If the loop completes without finding a match, return None

Program 5.1

```
def find_employee_by_id(employees, target_id):
    for employee in employees:
        if employee['id'] == target_id:
            return employee
    return None
```

Program 5:1

```
def find_employee_by_id(employee, target_id):
    for employee in employees:
        if employee['id'] == target_id:
            return employee
    return None

# Test the function
employees = [
    {'id': 1, 'name': 'Alice', 'department': 'HR'},
    {'id': 2, 'name': 'Bod', 'department': 'Engineering'},
    {'id': 3, 'name': 'charlie', 'department': 'Sales'}
```

output :-

{'id': 2, 'name': 'Bob', 'department': 'Engineer',

52) 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. The school needs to generate a report that displays students' scores. The school needs to generate a report that displays students' score in ascending order. Your task is to implement a feature that sorts the student record by their scores using the bubble sort algorithm

Algorithm :-

1. Initialization:
   - Get the length of the students list and store it in n.
2. Outer Loop
   - Iterate from i=0 to n-1. 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 pass.
4. Inner Loop:-
   - Iterate from j=0 to n-i-1. This loop compares adjacent elements in the list and performs swaps if necessary.
5. Compare and Swap
   - Compare and swap the score values
   - If students[j]['score] > students[j+1]['score] swap the two elements.

2) Completion
:) The function modifies the student list in place by sorting it by score.

Program 5.2

```
def bubble_sort_scores(students):
    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], students[j+1] = students[j+1],
                students[j]
                swapped = True
        # If no two elements were swapped
        the list is already sorted
        if not swapped:
            break
# Example usage
students = [
    {'name': 'Alice', 'score': 88},
    {'name': 'Bob', 'score': 95},
    {'name': 'Charlie', 'score': 75},
    {'name': 'Diana', 'score': 85}
]
print("Before sorting:")
for student in students:
    print(student)
print("\nAfter sorting:")
for student in students:
    print(student)
```

~~Out~~ ~~out~~

Before Sorting:

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

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

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

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

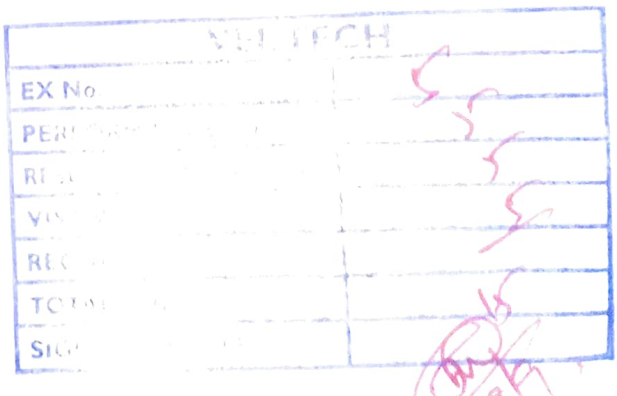
After Sorting:

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

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

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

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



Result: Thus the program for various searching and sorting operations is executed and verified Successfully