Date:- 20/8/25

Task-5

Implement various searching and sorting operations in Python.

a) Library Book Search (Linear Search) Binary Search).

Aim:- To write python program that allows searching for a book in a list of book titles using Linear search and Binary search (If the list of sorted).

Algorithm:-

1) Start from the first element of the list
2) compare each element with the search key.
3) If found, return the location.
4) If not found after checking all elements, return "Not found".

Binary search (for sorted lists)

1. Set low = 0 and high = n-1.
2. Find mid = (low + high)/2.
3) If key == list[mid], return location.
4) If key < list[mid], search the left half (high = mid-1).
5) else, search the right half (low = mid+1)
6) Repeat until found or list is exhausted

Program :-

```
#Linear Search
def linear_search(books, key):
    for i in range(len(books)):
        if books[i] == key:
            return i
    return -1

# binary search
def binary_search(books, key):
    low, high = 0, len(books)-1
    while low <= high:
        mid = (low + high)//2
        if books[mid] == key:
            return mid
        if book[mid] < key:
            low = mid + 1
        else:
            high = mid - 1
    return -1

#main
books = ["Python", "C", "Java", "HTML", "SQL"]
search_book = "Java"
print("Linear Search Result:")
pos = linear_search(books, search_book)
print("Binary Search Result:")
```

Simple output

Linear Search Result:
Book found at position 4
Sorted books: ['C', 'HTML', 'Java', 'Python', 'SQL']
Binary Search Result:
Book found at position 2.

```
if pos != -1:
    print(f"Book found at position {pos}")
else:
    print("Book not found")
```

Task-5(b)

Student grand organizer (sorting using insertion/selection).

Aim:- To write a python program that organized students' grades using different sorting techniques.

Algorithm:-

Bubble sort (Ascending)

1. Repeat for n-1 passes.
2. compare adjacent elements.
3. Swap if out of order.
4. After each pass the large element moves to the end.

Selection sort (Descending)

1. Find the maximum element in the unsorted part.
2. Swap with first element of the unsorted part.
3. Repeat until sorted.

Program :-

```
# Bubble sort (Ascending)
def bubble_sort(arr):
    n = len(arr)
    for i in range(n-1):
        for j in range(n-1-i):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] =
arr[j+1], arr[j]
    return arr
# selection sort (Descending)
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        max_idx = i
        for j in range(i+1, n):
            if arr[j] > arr[max_idx]:
                max_idx = j
        arr[i], arr[max_idx] =
arr[max_idx], arr[i]
    return arr
# main
grades = [88, 92, 75, 66, 90, 58, 99]
print("Original grades:", grades)
# Bubble sort (Ascending)
asc = bubble_sort(grades.copy())
```

Sample output

original arrays: (85, 92, 75, 66, 90, 58, 99)

Ascending (Bubble sort): (58, 66, 75, 85, 90, 92, 99)

Descending (Selection sort): (99, 92, 90, 85, 75, 66, 58)

Top 3 scores: [99, 92, 90]

print("Descending (Selection sort):", desc)

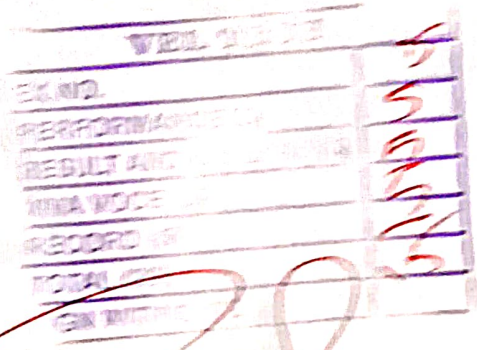
Selection sort (Descending)

desc_selection = sort(grades, reverse=True)

print("Descending (selection sort):", desc)

Top 3 scores

print("Top 3 scores:", desc[:3])



Results: Hence, the student grade report is sorted both ascending & descending order using different algorithms are executed successfully.