20/8/25 Task-5:- Implement various Searching and sorting operations In Python Programming.

(a). Library Book Search

Aim:- To search for a book in a library catalog using linear search for unsorted lists and binary search for sorted lists.

Algorithm:-

1. Linear search:
- check each book in the list one by one.
- If found, return its position
- If not found after checking all books, return -1.

2. Binary search
- Requires the list to be sorted.
- compare the target with middle element.
- If equal, return the position.
- If target is smaller, search the left half.
- If target is larger, search the right half.
- Repeat until found or search space exhausted.

Program:-

```
# Library Book search
def linear_search(books, target):
    for i in range(len(books)):
        if books[i] == target:
            return i
    return -1
def binary_search(books, target):
    low, high = 0, len(books)-1
    while low<high:
```

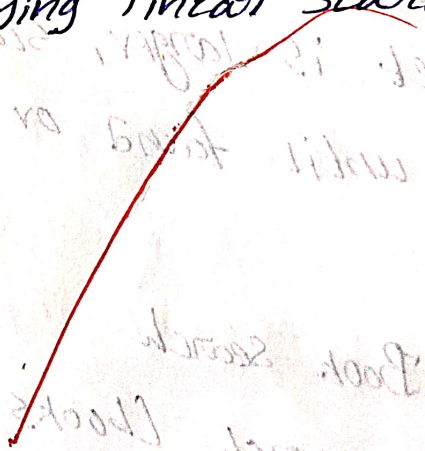
Output:-

Library Books: ['Python', 'Java', 'C++', 'JavaScript', 'HTML']

List is Sorted : False

Enter book to search: C++

Book found at position 3 using linear search.



```
        mid = (low + high) // 2
        if books[mid] == target:
            return mid
        elif books[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
        return -1
books = ["Python", "Java", "C++", "JavaScript", "HTML"]
print("Library Books:", books)
is_sorted = books == sorted(books)
print("List is sorted:", is_sorted)
target = input("Enter book to search:")
if is_sorted:
    result = binary_search(books, target)
    method = "Binary Search"
else:
    result = linear_search(books, target)
    method = "Linear search"
if result != -1:
    print("Books found at position {result+1} using {method}")
else:
    print("Book not found").
```

Result:-

The program to implement both search algorithms, to detect that the list was not sorted and used linear search to find "C++" at position 3 was successfully executed.

2018/25

(6) Student Grade Organizer

Aim:- To sort student grades using different algorithms And display the top 3 scores.

Algorithm:-

1. **Bubble sort (Ascending):**
   - Compare adjacent elements & swap it in wrong order
   - Repeat until no more swaps are needed.

2. **Selection sort (Descending):**
   - Find the max element and swap it first position.
   - Repeat for remaining elements.

3. **Top 3 Scores:**
   - After Sorting in descending order, the first three Elements are the top scores.

Program:-

```
def bubble_sort_asc(grades):
    n=len(grades)
    for i in range(n):
        for j in range(0, n-i-1):
            if grades[j] > grades[j+1]:
            grades[j], grades[j+1] = grades[j+1], grades[j]
    return grades
def selection_sort_desc(grades):
    n= len(grades)
    for i in range(n):
        max_idx =i
        for j in range(i+1, n):
            if grades[j] > grades[MAX_idx]:
                max_idx = j
```

Output :-

original Grades : [85, 92, 78, 90, 65, 88, 72]

Ascending order (Bubble sort) : [65, 72, 78, 85, 88, 90, 92]

Descending order (selection sort) : [92, 90, 88, 85, 78, 72, 65]

Top 3 scores : [92, 90, 88]

```
grades[i], grades[max_idx] = grades[max_idx], grades[i]

grades = [85, 92, 78, 90, 65, 88, 72]
Print ("original grades:", grades)
asc_sorted = bubble_sort_asc(grades.copy())
desc_sorted = selection_sort_desc(grades.copy())
Print ("Ascending order (Bubble sort):", asc_sorted)
Print ("Descending order (selection sort):", desc_sorted)
Print ("Top 3 scores:", desc_sorted[:3])
```

VEL TECH - CSE	
S NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	25
SIGN WITH DATE	

Result:-

Thus the student grade organizer in both ascending and descending order using different algorithms are executed successfully.

27/9/25