Date:- 20/8/25

Task-5

Implement various searching and sorting operations in python.

a.) Libary 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 position.
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 position.
4.) If key < list[mid], search the left half (right = 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[mind ) < key:
             low =mid +1
         else:
          high = mid -1
          return-1

#main
   boops = ["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 positiond
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 ubb (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 adjecent elements.
3. swap If out of order.
4. After each pass the largest element moves to the end.

selection sort (Desending)

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]
     retun arr
# maen
  grades = [88, 92, 75, 66, 90, 58, 99]
  Print("original grades :", grades)
  # Bubble sort(Ascending)
  asc= bubble_sort(grades.copy())
```

Sample output

original grades:(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,

Top 3 scores: [99, 92, 90]

```
print("Ascending (Bubble sort):", asc)
# selection sort (Descending)
desc = selection_sort(grades.copy())
print("Descending (selection sort):", desc)
# Top 3 scores
print("Top 3 scores:", desc[:3])
```

Result: Hence, the student grade organizer for both ascending & descending order using different algorithms are executed successfully.