Algorithms & Data Structures

Lab 10 (Sorting Algorithms in Python)

EX1: (Insertion algorithm)

```
# Function to implement insertion sort
def insertion_sort(arr):
    # Traverse through 1 to len(arr)
    for i in range(1, len(arr)):
        key = arr[i]
        # Move elements of arr[0..i-1], that are
        # greater than key, to one position ahead
        # of their current position
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key


# Driver code to test above
s = int(input('Enter the size if the list: '))
user_list = [0]*s
print('Enter the unsorted list.')
for index in range(s):
    user_list[index] = int(input('Enter a number: '))
print('The list before sorting: ', user_list)
insertion_sort(user_list)
print('The list after sorting: ', user_list)
```

EX2: (Selection algorithm)

```
# Function to implement selection sort
def selection_sort(arr):
    # Traverse through all array elements
    for i in range(len(arr)):
        # Find the minimum element in remaining
        # unsorted array
        min_idx = i
        for j in range(i + 1, len(arr)):
            if arr[min_idx] > arr[j]:
                min_idx = j
        # Swap the found minimum element with
        # the first element
        arr[i], arr[min_idx] = arr[min_idx], arr[i]


# Driver code to test above
s = int(input('Enter the size if the list: '))
user_list = [0]*s
print('Enter the unsorted list.')
for index in range(s):
    user_list[index] = int(input('Enter a number: '))
print('The list before sorting: ', user_list)
selection_sort(user_list)
print('The list after sorting: ', user_list)
```

EX3: (Bubble algorithm)

```
# Function to implement Bubble Sort
def bubble_sort(arr):
    n = len(arr)
    # Traverse through all array elements
    for i in range(n):
        # Last i elements are already in place
        for j in range(0, n - i - 1):
            # traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1],
arr[j]


# Driver code to test above
s = int(input('Enter the size if the list: '))
user_list = [0]*s
print('Enter the unsorted list.')
for index in range(s):
    user_list[index] = int(input('Enter a number: '))
print('The list before sorting: ', user_list)
bubble_sort(user_list)
print('The list after sorting: ', user_list)
```

EX4: (Merge algorithm)

```
# Function to implement Merge Sort
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2  # Finding the mid of the
array
        L = arr[:mid]  # Dividing the array elements
        R = arr[mid:]  # into 2 halves

        merge_sort(L)  # Sorting the first half
        merge_sort(R)  # Sorting the second half

        i = j = k = 0

        # Copy data to temp arrays L[] and R[]
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1

        # Checking if any element was left
        while i < len(L):
            arr[k] = L[i]
            i += 1
```

```
            k += 1

        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1


# Driver code to test above
s = int(input('Enter the size if the list: '))
user_list = [0]*s
print('Enter the unsorted list.')
for index in range(s):
    user_list[index] = int(input('Enter a number: '))
print('The list before sorting: ', user_list)
merge_sort(user_list)
print('The list after sorting: ', user_list)
```

EX5: (Quick algorithm)

```
# This function takes last element as pivot, places
# the pivot element at its correct position in sorted
# array, and places all smaller (smaller than pivot)
# to left of pivot and all greater elements to right
# of pivot


def partition(arr, low, high):
```

```
    i = (low - 1)  # index of smaller element
    pivot = arr[high]  # pivot

    for j in range(low, high):

        # If current element is smaller than the
pivot
        if arr[j] < pivot:
            # increment index of smaller element
            i = i + 1
            arr[i], arr[j] = arr[j], arr[i]

    arr[i + 1], arr[high] = arr[high], arr[i + 1]
    return (i + 1)


# The main function that implements QuickSort
# arr[] --> Array to be sorted,
# low  --> Starting index,
# high  --> Ending index

# Function to implement Quick sort
def quick_sort(arr, low, high):
    if low < high:
        # pi is partitioning index, arr[p] is now
        # at right place
        pi = partition(arr, low, high)
```

```
        # Separately sort elements before
        # partition and after partition
        quick_sort(arr, low, pi - 1)
        quick_sort(arr, pi + 1, high)


# Driver code to test above
s = int(input('Enter the size if the list: '))
user_list = [0]*s
print('Enter the unsorted list.')
for index in range(s):
    user_list[index] = int(input('Enter a number: '))
print('The list before sorting: ', user_list)
quick_sort(user_list, 0, s-1)
print('The list after sorting: ', user_list)
```