ASSIGNMENT - 5

Name : B. Ravi kumar

Reg no : 192311246

course : Data structure for stack overflow.

Course Code : CSA0389.

Write the algorithm for insertion sort and sort the following sequence :

3, 1, 4, 1, 5, 9, 2, 6, 5.

20) Algorithm for Insertion :

(i) Begin with the second element in the first.

(ii) Compare the current element to the previous elements.

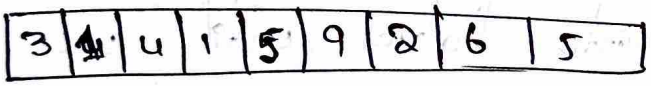
(iii) Shift all larger elements one position the right.

(iv) Insert the current elements into its correct position.

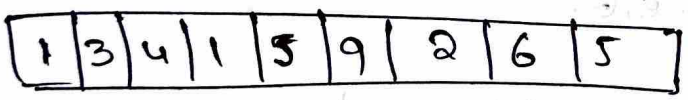
(v) Repeat steps for each element :

Sorting the sequence :

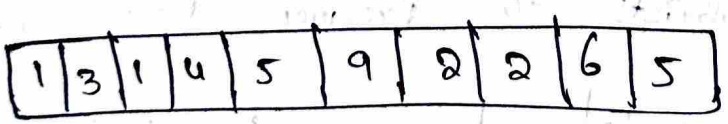
3, 1, 4, 1, 5, 9, 2, 6, 5.



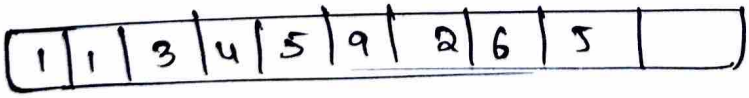
Compare 3 & 1, 3 > 1
Swap 3, 1.



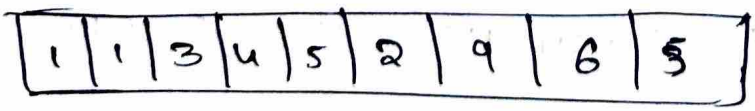
Compare 4 & 1, 4 > 1
Swap 4, 1.



Compare 3 & 1, 3 > 1
Swap 3, 1



Compare 9 & 2, 9 > 2,
Swap 9, 2.



Compare 5 & 2, 5 > 2
Swap 5, 2.

1	1	3	4	2	5	9	6	5

Compare 4 & 2, 4 > 2

Swap 4, 2.

1	1	3	2	4	5	9	6	5

Compare 3 & 2, 3 > 2.

Swap 3, 2.

1	1	2	3	4	5	9	6	5

Compare 9 & 6, 9 > 6

Swap 9, 6.

1	1	2	3	4	5	6	9	5

Compare 9 & 5, 9 > 5

Swap 9, 5

1	1	2	3	4	5	6	5	9

Compare 6 & 5, 6 > 5

Swap 6, 5

1	1	2	3	4	5	5	6	9

Sorted.

Sorted sequence : 1, 1, 2, 3, 4, 5, 5, 6, 9.

Merge sort Procedure :

* split the list into halves until each sublist has one element.

* Combine the sublists to Produce new sorted sorted sublists until there is one sorted list.

Sorted list : 0, 1, 8, 27, 64, 125, 216, 343, 512, 729

Draw the concept map of partitioning in quick sort try to write an algorithm for it, which is as follows & develop a program. considering the steps.

Ans: Algorithm :

* select the element at the highest index as the pivot.
* set 'left' to the low index and 'right' to the high index -1.
* move 'left' rightwards and 'right' leftwards until 'left' is greater than or equal to 'right' swapping elements as the needed.
* swap the pivot with the element at the 'left' pointer position.
* Return the index of pivot element.

Program :

```
#include <stdio.h>
int main () {
    int arr[] = {64, 8, 216, 512, 27, 729, 0, 1, 343,
                 125};
    int n = size of (arr) / size of (arr[0]);
    int low = 0, high = n-1;
```

```
while (left <= right) {
    int pivot = arr[high];
  int left = low;
   int right = high - 1;
  while (left <= right) {
    while (left <= right && arr[left] < pivot) {
    left++;
}
while (right >= low && arr[right] > pivot) {
    right--;
}
if (left < right)
  {
    int temp = arr[left];
     arr[left] = arr[right];
      arr[right] = temp;
       left++;
    right--;
  }
}
```

```
int temp = arr[left];
    arr[left] = arr[high];
    arr[high] = temp;
        high = left - 1;
    if (high < low) {
        low = left + 1;
        high = n-1;
    }
}
printf("Sorted array:");
for (int i= 0; i<n; i++) {
    printf("%d ", arr[i]);
}
printf("\n");
return 0;
}
```

output:

Sorted array : 0, 1, 8, 27, 64, 125, 216, 343, 512,

729