**Algorithm-1**

**Algorithm of store data of 5 student in array and display it.**

1. Start

2. id[] as student id, name[] as studentname [], address [] as address store value via for loop

3. i=1 as int

4. check if i<=5 true continue or else go to step 6

5. Print id=id[], studentname=studentname[],Address=Address[]

6. i=i+1

7. End

**Algorithm-2**

**Bubble Sort**

1. Start

2. make array employeeid[ ] and store via for loop

3. for all the element of employeeid[]

If employeeid[ ]>employeeid[i=1]

t=employeeid[i];

employeeid[i] = employeid [j]

employeeid[j]=t

end if

end for

4. print employeeid[]

5. end

**Algorithm – 3**

**Merge Sort Algorithm**

1. Start

2. Read last and first as int

3. Make array employeeid[ ] and store via for loop

4. Find the middle index of array middle 1+(last-first)/2

5. Divide the array from middle

6. Call merge sort for the first half of the array mergesort(employeeid,first,middle)

7. Call merge sort for the second half of the array mergesort(employeeid,middle+1,last)

8. merge the two sorted half into a single sorted array

9. End

**Algorithm - 4**

**Selection sort Algorithm**

1. Start
2. Set MINEemployeeid to location 0
3. Search the minimum employeeid in the list
4. Swap with value at location MINEemployeeid
5. Increment MINEemployeeid to point to the next employeeid
6. Repeat until list is sorted
7. End

**Algorithm – 5**

**Quick Sort Algorithm**

1. Start

2**.** Choose the highest index value has employeId has pivot

3. Take two employeId to point left and right of the list

4. Left employeId to the low index

5. Right employeId to the high index

6. While employeId at left is less than pivot move right.

7. While employeId at right is greater than pivot move left.

8. If both step 6 an 7 doesn’t match swap left and right.

9. If left >= right, the point where they met is new pivot

10. End

**Algorithm – 6**

**Sort a stack using employeId**

1. Start

2. Initialize array employeId [ ] of size 5

3. Create a stack and push all the elements of array employeid[ ] in stack.

4. Create Another temp stack.

5. Poop first elements from original stack and store in temp.

6. If temp > pop elements than push pop elements in temp stack. Otherwise push temp variable.

7. Repeat step 5 and step 6 until stack is sorted.

8. End