Experiment : 2

```
// Merge Sort

#include <stdio.h>


void merge(int arr[], int p, int q, int r) {


  int n1 = q - p + 1;
  int n2 = r - q;

  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)
    L[i] = arr[p + i];
  for (int j = 0; j < n2; j++)
    M[j] = arr[q + 1 + j];


  int i, j, k;
  i = 0;
  j = 0;
  k = p;
  while (i < n1 && j < n2) {
    if (L[i] <= M[j]) {
      arr[k] = L[i];
      i++;
    } else {
      arr[k] = M[j];
      j++;
    }
    k++;
  }


  while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
  }

  while (j < n2) {
    arr[k] = M[j];
    j++;
    k++;
  }
}
```

```
void mergeSort(int arr[], int l, int r) {
  if (l < r) {


    int m = l + (r - l) / 2;

    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);


    merge(arr, l, m, r);
  }
}


void printArray(int arr[], int size) {
  for (int i = 0; i < size; i++)
    printf("%d ", arr[i]);
  printf("¥n");
}


int main() {
         int arrSize;
    scanf("%d", &arrSize);

    int arr[arrSize];

    for (int i=0; i<arrSize; ++i)
        scanf("%d", &arr[i]);

    mergeSort(arr, 0, arrSize - 1);

    printf("Sorted array: ¥n");
    printArray(arr, arrSize);
}


// Quick Sort

#include <stdio.h>

void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}
```

```
}

int partition(int *arr, int low, int high) {

  int pivot = arr[high];

  int i = (low - 1);

  for (int j = low; j < high; j++) {
    if (arr[j] <= pivot) {

      i++;

      swap(&arr[i], &arr[j]);
    }
  }

  swap(&arr[i + 1], &arr[high]);

  return (i + 1);
}
```

```
void quickSort(int *arr, int low, int high) {
  if (low < high) {


    int pi = partition(arr, low, high);


    quickSort(arr, low, pi - 1);


    quickSort(arr, pi + 1, high);
  }
}


void printArray(int *arr, int size) {
  for (int i = 0; i < size; ++i) {
    printf("%d ", arr[i]);
  }
  printf("¥n");
}


int main() {
         int arrSize;
         scanf("%d", &arrSize);

         int arr[arrSize];

         for (int i=0; i<arrSize; ++i)
             scanf("%d", &arr[i]);

         quickSort(arr, 0, arrSize - 1);

         printf("Sorted array: ¥n");
         printArray(arr, arrSize);
}
```