**Experiment No.8**

**Aim:** To Implement Merge sort and Heap sort.

**Code:**

**#include <stdio.h>**

**void printArray(int \*A, int n)**

**{**

**for (int i = 0; i < n; i++)**

**{**

**printf("%d ", A[i]);**

**}**

**printf("\n");**

**}**

**void merge(int A[], int mid, int low, int high)**

**{**

**int i, j, k, B[100];**

**i = low;**

**j = mid + 1;**

**k = low;**

**while (i <= mid && j <= high)**

**{**

**if (A[i] < A[j])**

**{**

**B[k] = A[i];**

**i++;**

**k++;**

**}**

**else**

**{**

**B[k] = A[j];**

**j++;**

**k++;**

**}**

**}**

**while (i <= mid)**

**{**

**B[k] = A[i];**

**k++;**

**i++;**

**}**

**while (j <= high)**

**{**

**B[k] = A[j];**

**k++;**

**j++;**

**}**

**for (int i = low; i <= high; i++)**

**{**

**A[i] = B[i];**

**}**

**}**

**void mergeSort(int A[], int low, int high){**

**int mid;**

**if(low<high){**

**mid = (low + high) /2;**

**mergeSort(A, low, mid);**

**mergeSort(A, mid+1, high);**

**merge(A, mid, low, high);**

**}**

**}**

**int main()**

**{**

**int A[] = {12, 11, 13, 6, 5, 7};**

**int n = 6;**

**printArray(A, n);**

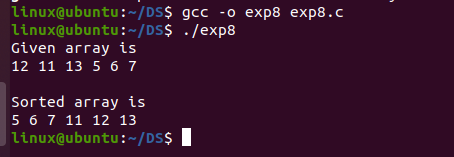
**mergeSort(A, 0, 5);**

**printArray(A, n);**

**return 0;**

**}**

**OUTPUT**:



**CODE: HEAP SORT**

#include **<stdio.h>**

/\* function to heapify a subtree. Here 'i' is the

index of root node in array a[], and 'n' is the size of heap. \*/

void heapify(int a[], int n, int i)

{

    int largest = i; // Initialize largest as root

    int left = 2 \* i + 1; // left child

    int right = 2 \* i + 2; // right child

    // If left child is larger than root

    if (left **<** **n** && a[left] **>** a[largest])

        largest = left;

    // If right child is larger than root

    if (right **<** **n** && a[right] **>** a[largest])

        largest = right;

    // If root is not largest

    if (largest != i) {

        // swap a[i] with a[largest]

        int temp = a[i];

        a[i] = a[largest];

        a[largest] = temp;

        heapify(a, n, largest);

    }

}

/\*Function to implement the heap sort\*/

void heapSort(int a[], int n)

{

    for (int i = n / 2 - 1; i **>**= 0; i--)

        heapify(a, n, i);

    // One by one extract an element from heap

    for (int i = n - 1; i **>**= 0; i--) {

        /\* Move current root element to end\*/

        // swap a[0] with a[i]

        int temp = a[0];

        a[0] = a[i];

        a[i] = temp;

        heapify(a, i, 0);

    }

}

/\* function to print the array elements \*/

void printArr(int arr[], int n)

{

    for (int i = 0; i **<** **n**; ++i)

    {

        printf("%d", arr[i]);

        printf(" ");

    }

}

int main()

{

    int a[] = {48, 10, 23, 43, 28, 26, 1};

    int n = sizeof(a) / sizeof(a[0]);

    printf("Before sorting array elements are - \n");

    printArr(a, n);

    heapSort(a, n);

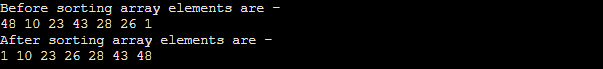
    printf("\nAfter sorting array elements are - \n");

    printArr(a, n);

    return 0;

}

OUTPUT:



**Conclusion:**

The working of divide and conquer method along with two different algorithms such as Merge Sort Algorithms and Quick Sort Algorithm have been understood where quick sort is more efficient and works faster than merge sort in case of smaller array size or datasets .