Assignment - 3 :

a) WAP to implement HEAP sort.

b) WAP to implement COUNTING sort.

(a) Heap Sort –

Algorithm –

Swap Algorithm –

Procedure: swap(*a, *b)
Input: a – a pointer to an address
b – a pointer to another address
Output: a and b are addresses that interchanges.

1. temp ← *a.
2. *a ← *b.
3. *b ← temp.
4. return.

Heap Sort Algorithm –

Procedure: heapSort(arr, n)
Input: arr – An array containing elements to be sorted.
n – Size of array arr.
Output: A – Sorted Array

1. for i ← ⌊ n/2 ⌋ – 1 downto 0 then
2. call heapify(arr,n,i).
3. end for.
4. for i ← n – 1 downto 0 then
5. call swap(&arr[0], &arr[i]).
6. call heapify(arr,n,0).
7. end for.
8. return.

Heapify Algorithm –

Procedure: heapify(arr, n, i)

Input: arr – A rearranged array arr.

n – any index of array arr.

i – any index of array arr.

Output: Heapify the root element again so that we have the highest element at root.

1. largest ← i.
2. left ← 2*i+1.
3. right ← 2*i+2.
4. if left < n and arr[left] > arr[largest] then
5. largest ← left.
6. end if.
7. if right < n and arr[right] > arr[largest] then
8. largest ← right.
9. end if.
10. if largest != i then
11. call swap(&arr[i], &arr[largest]).
12. call heapify(arr, n, largest).
13. end if.
14. return.

Program –

```c
#include <stdio.h>

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

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
    return;
}

void heapify(int arr[], int n, int i)
{
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && arr[left] > arr[largest])
        largest = left;

    if (right < n && arr[right] > arr[largest])
        largest = right;

    if (largest != i)
    {
        swap(&arr[i], &arr[largest]);
        heapify(arr, n, largest);
    }
    return;
}

void heapSort(int arr[], int n)
{
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    for (int i = n - 1; i >= 0; i--)
    {
        swap(&arr[0], &arr[i]);
        heapify(arr, i, 0);
    }
    return;
}

int main()
{

    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    int A[n];
    for (int i = 0; i < n; i++)
    {
        printf("Enter array element (%d): ", i + 1);
        scanf("%d", &A[i]);
    }

    printf("Unsorted array: ");
    printArray(A, n);
    heapSort(A, n);
    printf("Sorted array: ");
    printArray(A, n);

    return 0;
}
```

Output –

```
Falguni Sarkar@MELOPHILE  G:\Semester~4\Design & Analysis of Algorithm\Lab\Assignment 3
❯ cd "g:\Semester~4\Design & Analysis of Algorithm\Lab\Assignment 3\" ; if ($?) { gcc Heap_Sort.c -o Heap_Sort } ; if ($?) { .\Heap_Sort }
Enter the size of the array: 8
Enter array element (1): -5
Enter array element (2): -10
Enter array element (3): 0
Enter array element (4): 3
Enter array element (5): 8
Enter array element (6): 5
Enter array element (7): -1
Enter array element (8): 10
Unsorted array: -5 -10 0 3 8 5 -1 10
Sorted array: -10 -5 -1 0 3 5 8 10
```

(b) Counting Sort –

Algorithm –

Procedure: countSort(A, size)
Input: A – An array containing elements to be sorted.
size – No. of elements of array A.
Output: A – Sorted Array

1. max ← INT_MIN.
2. min ← INT_MAX.
3. for i ← 0 to size – 1 do
4. if max < A[i] then
5. max ← A[i].
6. end if.
7. If min > A[i] then
8. min ← A[i].
9. end if.
10. end for.
11. range ← max – min + 1.
12. for i ← 0 to range – 1 do
13. count[i] = 0.
14. end for.
15. for i ← 0 to size – 1 do
16. count[A[i]-min]++.
17. end for.
18. for i ← 1 to range – 1 do
19. count[i] += count[i - 1].
20. end for.
21. for i ← size - 1 downto 0 do
22. output[count[A[i] - min] - 1] = A[i];
23. count[A[i] - min]--;
24. end for.
25. for i ← 0 to size – 1 do
26. A[i] = output[i];
27. end for.
28. return.

Program –

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

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

void countSort(int A[], int size)
{
    int maxSize, i, max = INT_MIN, min = INT_MAX;

    for(int i = 0; i < size; i++)
    {
        if(max < A[i])
            max = A[i];
        if(min > A[i])
            min = A[i];
    }

    int range = max-min+1, count[range], output[size];

    for(i = 0; i < range; i++)
        count[i] = 0;

    for (int i = 0; i < size; i++)
        count[A[i] - min]++;

    for (int i = 1; i < range; i++)
        count[i] += count[i - 1];

    for (int i = size - 1; i >= 0; i--)
    {
        output[count[A[i] - min] - 1] = A[i];
        count[A[i] - min]--;
    }

    for (int i = 0; i < size; i++)
        A[i] = output[i];

    return;
}

int main()
{
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    int A[n];
    for (int i = 0; i < n; i++)
    {
        printf("Enter array element (%d): ", i + 1);
        scanf("%d", &A[i]);
    }

    printf("Unsorted array: ");
    printArray(A, n);
    countSort(A, n);
    printf("Sorted array: ");
    printArray(A, n);

    return 0;
}
```

Output –

```
Falguni Sarkar@MELOPHILE  G:\Semester~4\Design & Analysis of Algorithm\Lab\Assignment 3
❯ cd "g:\Semester~4\Design & Analysis of Algorithm\Lab\Assignment 3\" ; if ($?) { gcc Count_Sort.c -o Count_Sort } ; if ($?) { .\Count_Sort }
Enter the size of the array: 8
Enter array element (1): -5
Enter array element (2): -10
Enter array element (3): 0
Enter array element (4): 3
Enter array element (5): 8
Enter array element (6): 5
Enter array element (7): -1
Enter array element (8): 10
Unsorted array: -5 -10 0 3 8 5 -1 10
Sorted array: -10 -5 -1 0 3 5 8 10
Falguni Sarkar@MELOPHILE  G:\Semester~4\Design & Analysis of Algorithm\Lab\Assignment 3
❯ cd "g:\Semester~4\Design & Analysis of Algorithm\Lab\Assignment 3\" ; if ($?) { gcc Count_Sort.c -o Count_Sort } ; if ($?) { .\Count_Sort }
Enter the size of the array: 7
Enter array element (1): 3
Enter array element (2): 1
Enter array element (3): 9
Enter array element (4): 7
Enter array element (5): 1
Enter array element (6): 2
Enter array element (7): 4
Unsorted array: 3 1 9 7 1 2 4
Sorted array: 1 1 2 3 4 7 9
```