TUTORIAL XIII:
Heap Implementation
U19CS012 [D-12]

Implement the following operations in context to ***Heap Data Structure***:

1) Build Max Heap

2) Heapify Procedure

3) Insert a New Element in the Existing Heap

4.) Extract Max or Delete an Element from Max Heap

5.) Heap Sort

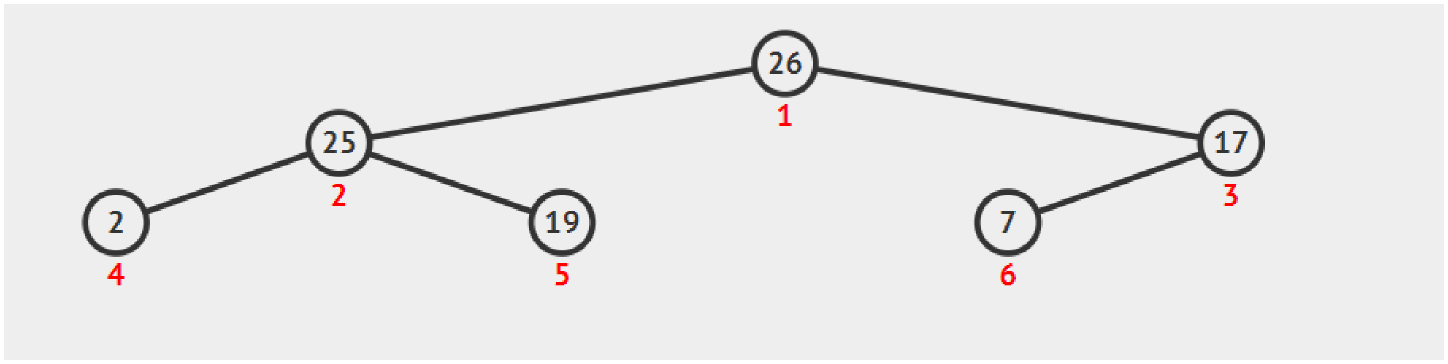
Code:

```
// Implementation Of Heap Operations
// a. Build max heap
// b. Heapify procedure
// c. Insert a new element in the existing heap
// d. Delete_Max max or delete an element from the max heap
// e. Heap Sort

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

// Defines the Maximum Size of Heap [Stored in Form Of Array]
#define MAX 1005

// Global Iterator
int i;

// Utility Function to Swap
void swap(int *x, int *y);

// Heapify Procedure to Comlpete Binary Tree to Heap
void heapify(int heap[], int n, int i);

// Utility Function for Insert
void create(int heap[], int n);

// To Display the Max Heap
void Display_Max_Heap(int heap[], int n);

// To Insert Element in Max-Heap
void Insert(int heap[], int *n, int val);
```

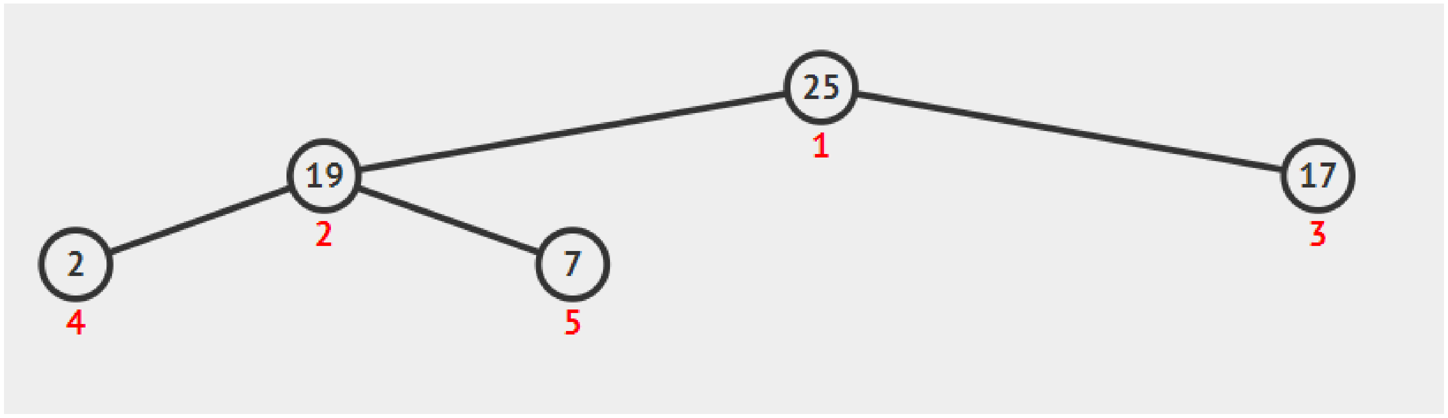
```c
// Delete the Maximum Element in Max Heap
void Delete_Max(int heap[], int *n);

// Function to Implement Heap Sort Algorithm
void Heap_Sort(int heap[], int n);

int main()
{
    int heap[MAX];
    int len = 0;

    int choice;
    printf("\nHEAP\n");

    printf(" 1 -> Insert a New Node in Heap\n");
    printf(" 2 -> Delete Element in Max-Heap\n");
    printf(" 3 -> Heap Sort\n");
    printf(" 4 -> Display Inorder Traversal of Max Heap\n");
    printf(" 5 -> Exit\n");
    int x;

    while (1)
    {
        printf("Enter your choice : ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter Node Value : ");
            scanf("%d", &x);
            Insert(heap, &len, x);
            break;
        case 2:
            Delete_Max(heap, &len);
            break;
        case 3:
            Heap_Sort(heap, len);
            break;
        case 4:
            Display_Max_Heap(heap, len);
            break;
        case 5:
            exit(0);
            break;
        default:
            printf("Enter a Valid Choice!");
            break;
        }
    }
    return 0;
}

// Utility Function to Swap
void swap(int *x, int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}

// Heapify Procedure to Comlpete Binary Tree to Heap
void heapify(int heap[], int n, int i)
{
    int l = 2 * i + 1;
    int r = 2 * i + 2;
    int large = i;
    if (l < n && heap[l] > heap[large])
    {
        large = l;
    }
    if (r < n && heap[r] > heap[large])
    {
        large = r;
    }
    if (i != large)
    {
        swap(&heap[i], &heap[large]);
        heapify(heap, n, large);
    }
}

// Utility Function for Insert
void create(int heap[], int n)
{
    for (i = n / 2 - 1; i >= 0; --i)
    {
        heapify(heap, n, i);
    }
}

// To Display the Max Heap
void Display_Max_Heap(int heap[], int n)
{
    printf("MAX HEAP : ");
    for (i = 0; i < n; ++i)
    {
        printf("%d ", heap[i]);
    }
    printf("\n");
}

// To Insert Element in Max-Heap
void Insert(int heap[], int *n, int val)
{
    *n = *n + 1;
    heap[*n - 1] = val;
    create(heap, *n);
}

// Delete the Maximum Element in Max Heap
void Delete_Max(int heap[], int *n)
{
    heap[0] = heap[*n - 1];
    *n = *n - 1;
    heapify(heap, *n, 0);
}

// Function to Implement Heap Sort Algorithm
void Heap_Sort(int heap[], int n)
{
    for (i = n - 1; i > 0; --i)
    {
        swap(&heap[0], &heap[i]);
        heapify(heap, i, 0);
    }
    for (i = 0; i < n / 2; ++i)
    {
        swap(&heap[i], &heap[n - 1 - i]);
    }
}
```

Test Cases:

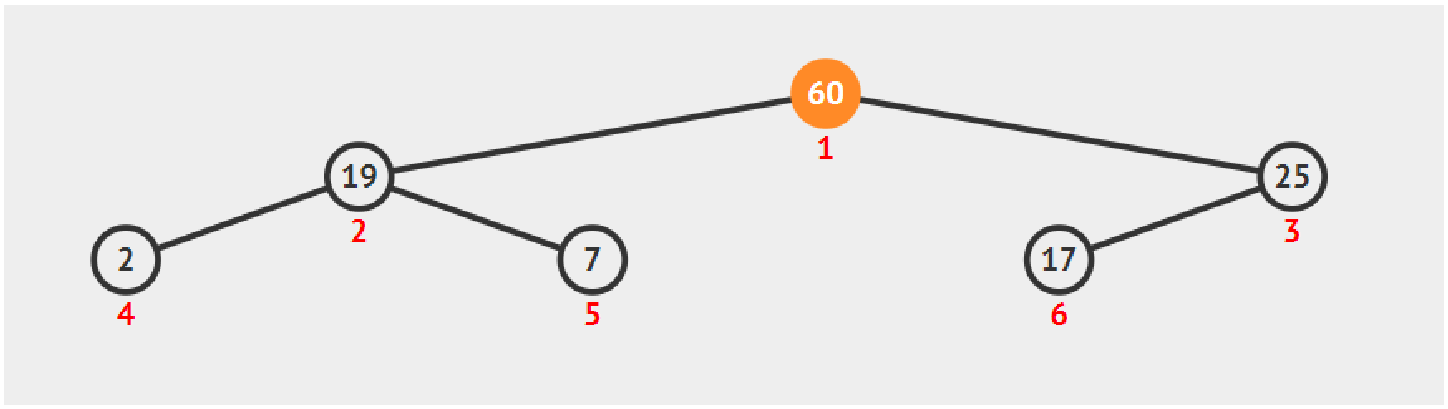
A.) Insertion in Max Heap to form below Heap



B.) After Extract Max or Delete Max Element



C.) After Inserting "60"



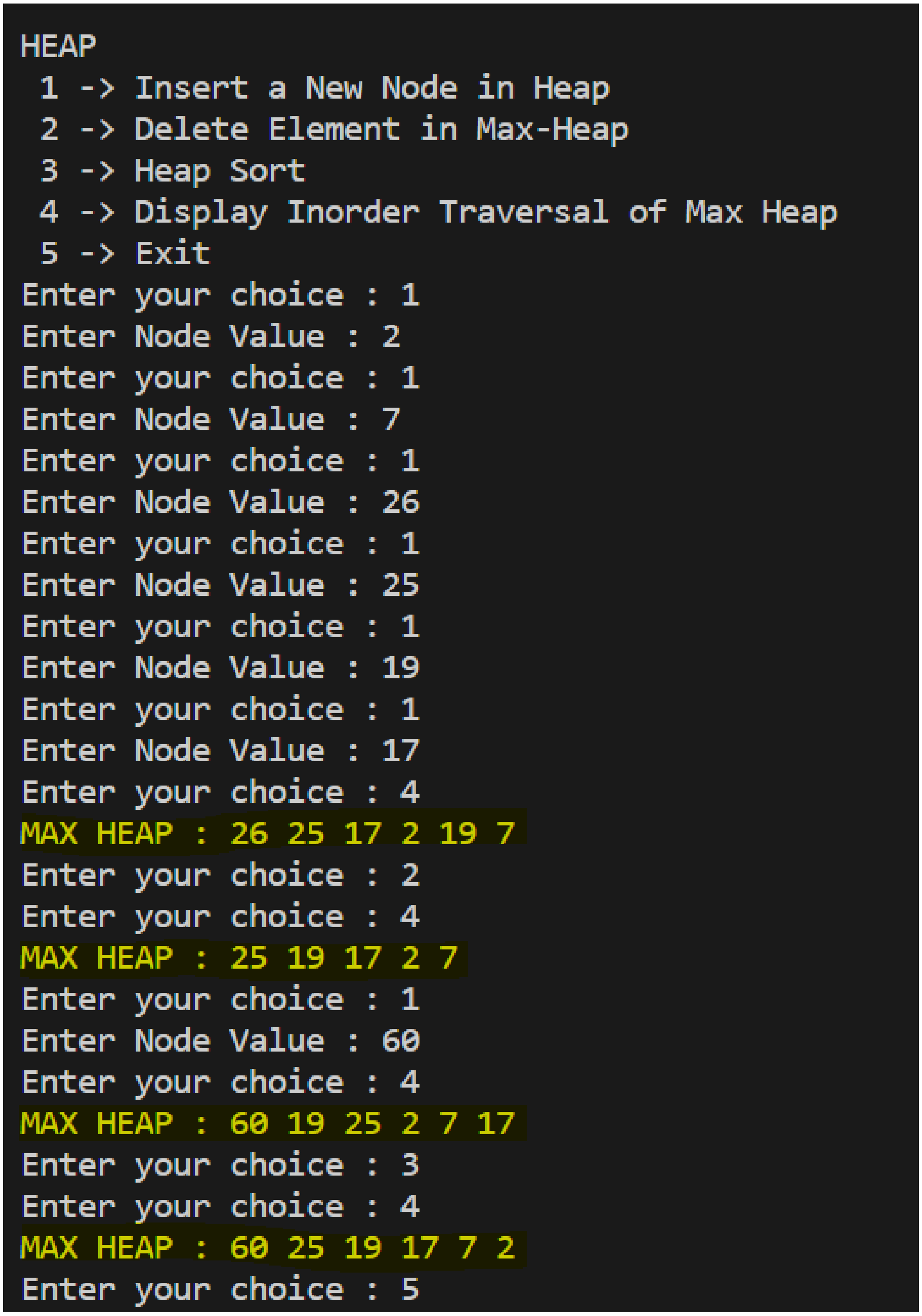
D.) In Heap Sort

It will Insert all Elements and Remove the Highest One-by-One.

Resulting in

"60 25 19 17 7 2"

Execution



```
HEAP
 1 -> Insert a New Node in Heap
 2 -> Delete Element in Max-Heap
 3 -> Heap Sort
 4 -> Display Inorder Traversal of Max Heap
 5 -> Exit
Enter your choice : 1
Enter Node Value : 2
Enter your choice : 1
Enter Node Value : 7
Enter your choice : 1
Enter Node Value : 26
Enter your choice : 1
Enter Node Value : 25
Enter your choice : 1
Enter Node Value : 19
Enter your choice : 1
Enter Node Value : 17
Enter your choice : 4
MAX HEAP : 26 25 17 2 19 7
Enter your choice : 2
Enter your choice : 4
MAX HEAP : 25 19 17 2 7
Enter your choice : 1
Enter Node Value : 60
Enter your choice : 4
MAX HEAP : 60 19 25 2 7 17
Enter your choice : 3
Enter your choice : 4
MAX HEAP : 60 25 19 17 7 2
Enter your choice : 5
```