**Task-4**

#include <iostream>

using namespace std;

// class representing node of a binary tree

class Node {

    public:

    int data;

    Node \*left;

    Node \*right;

    Node(int d) {

        data = d;

        left = right = NULL;

    }

};

// function to store the inorder traversal of tree in a list

int i =0;

void Inorder(Node \*root, int \*inOrderTraversal) {

    if (root != NULL) {

        Inorder(root->left, inOrderTraversal);

        cout<<root->data<<" ";

        inOrderTraversal[i] = root->data;

        i++;

        Inorder(root->right, inOrderTraversal);

    }

}

// to heapify a subtree with root at given index

void min\_heap(int arr[], int i, int N)

{

    int largest = i;

    int right = 2 \* i + 2;

    int left = 2 \* i + 1;

    if (left < N && arr[left] < arr[i])

        largest = left;

    if (right < N && arr[right] < arr[largest])

        largest = right;

    if (largest != i)

    {

        swap(arr[i], arr[largest]);

        min\_heap(arr, largest, N);

    }

}

// creating max heap

void build\_minHeap(int arr[], int N)

{

    for (int i = (N - 2) / 2; i >= 0; --i)

        min\_heap(arr, i, N);

}

//displaying array

void Display(int \*Arrr, int size)

{

    for (int i = 0; i < size; ++i)

        cout << Arrr[i] << " ";

}

void BST\_MinHead(int \*arr, int size){

    build\_minHeap(arr,i);

    cout<<"\nMin Heap\n";

    for(int i =0;i<size;i++){

        cout<<arr[i]<<" ";

     }

}

int main() {

    // Example Tree

    int\* arr = new int[100];

    Node\* root = new Node(23);

    root->left = new Node(15);

    root->right = new Node(18);

    root->left->left = new Node(65);

    root->left->right = new Node(9);

    root->right->left = new Node(13);

    root->right->right = new Node(23);

    cout << "Tree\n";

    Inorder(root, arr);

    BST\_MinHead(arr, i);

    return 0;

}

