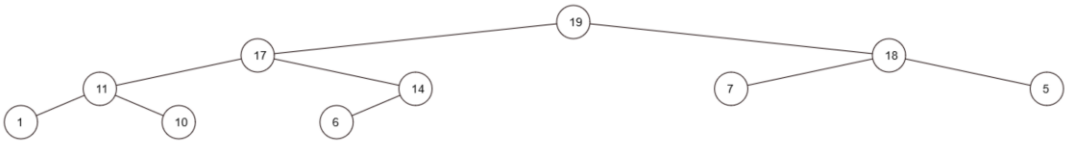
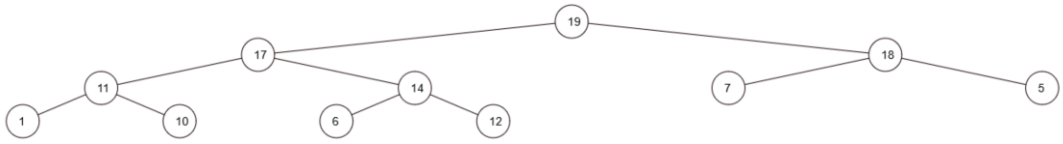
Exercise 5: Random list $S_2 = [1, 7, 18, 6, 14, 19, 5, 10, 11, 17]$

a) Draw the heap tree

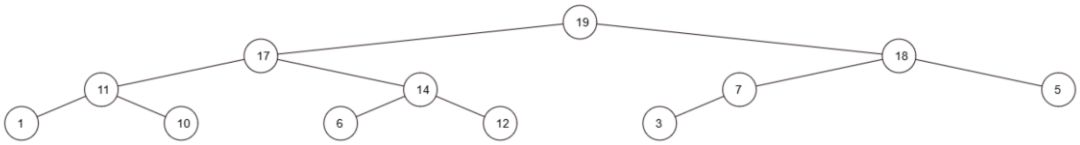


b) Insert elemts from $S_1 = [12, 3, 4, 15, 16, 13, 0, 20, 2, 9]$

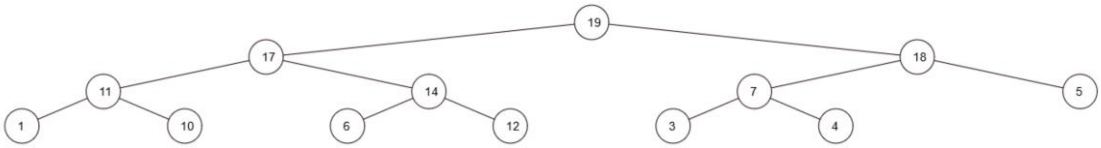
- Insert 12



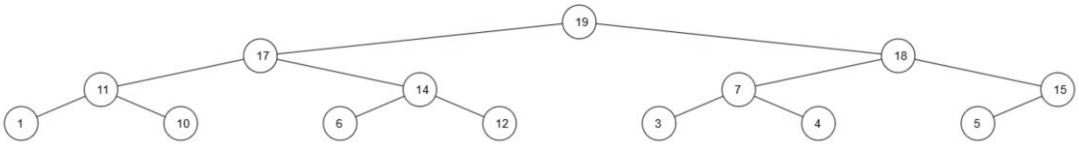
- Insert 3



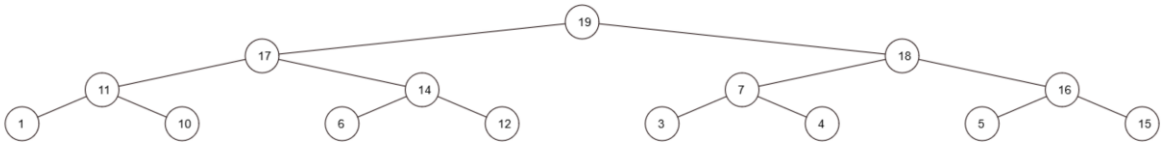
- Insert 4



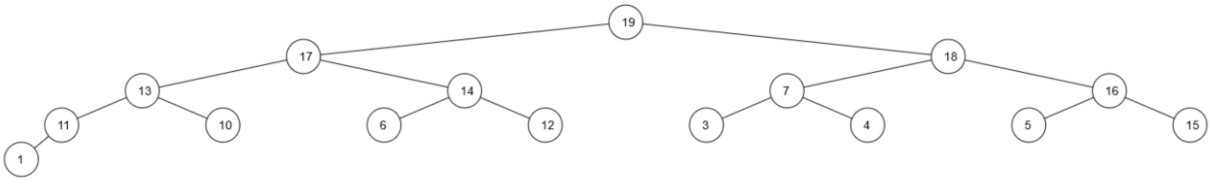
- Insert 15



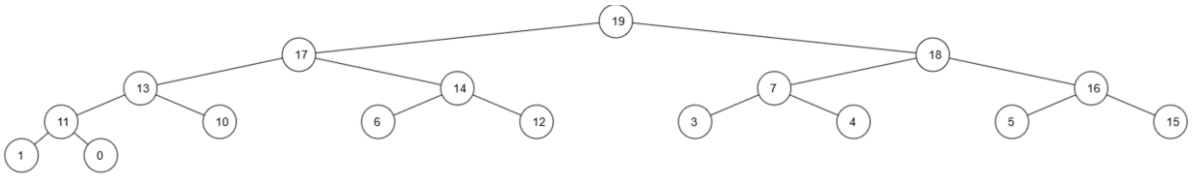
- Insert 16



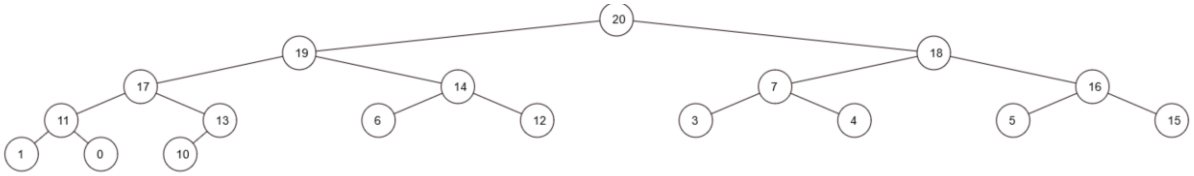
- Insert 13



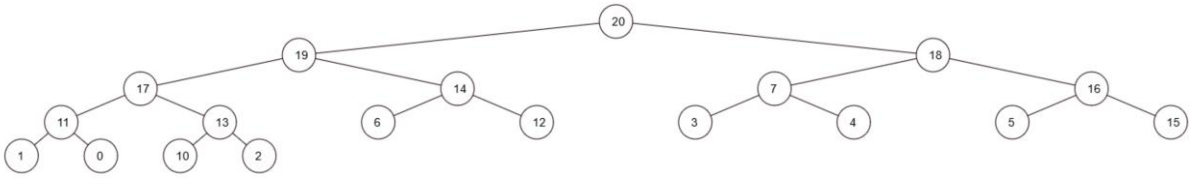
- Insert 0



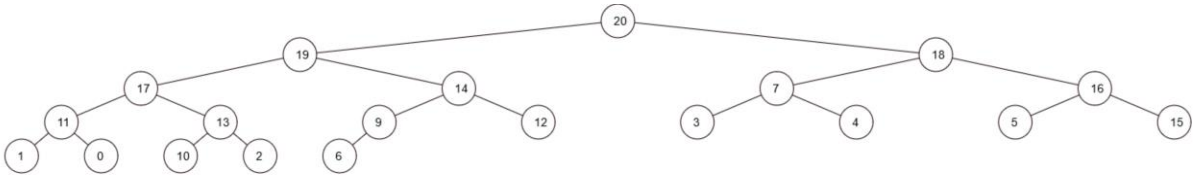
- Insert 20



- Insert 2



- Insert 9



c) Write out the procedure to find and remove the maxium element from heap tree in detail

```
void heapify(Node* &root) {
    if (root == nullptr) return;

    Node* largest = root;
    if (root->left != nullptr && root->left->value > largest->value) {
        largest = root->left;
    }
    if (root->right != nullptr && root->right->value > largest->value) {
        largest = root->right;
    }

    if (largest != root) {
        swap(root->value, largest->value);
```

```
        heapify(largest);
    }
}

Node* getRightmostLeaf(Node* root) {
    if (root == nullptr) return nullptr;
    while (root->right != nullptr) {
        root = root->right;
    }
    return root;
}

void removeRoot(Node* &root) {
    if (root == nullptr) return;

    if (root->left == nullptr && root->right == nullptr) {
        delete root;
        root = nullptr;
    } else if (root->left == nullptr) {
        Node* temp = root;
        root = root->right;
        delete temp;
    } else if (root->right == nullptr) {
        Node* temp = root;
        root = root->left;
        delete temp;
    } else {
        Node* rightmostLeaf = getRightmostLeaf(root->left);
        root->value = rightmostLeaf->value;
        removeRoot(rightmostLeaf);
        heapify(root);
    }
}
```