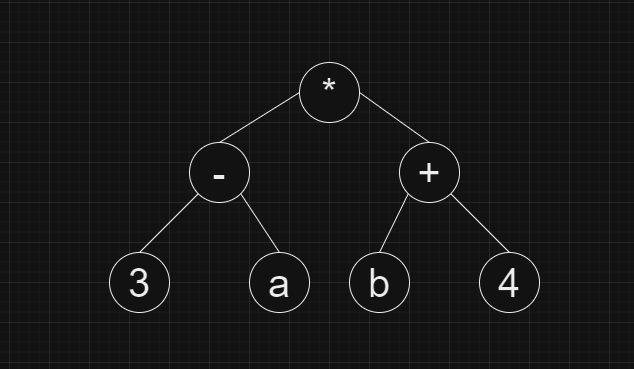
**Name:** Nguyễn Hữu Khang

**Student id number: 2011365**

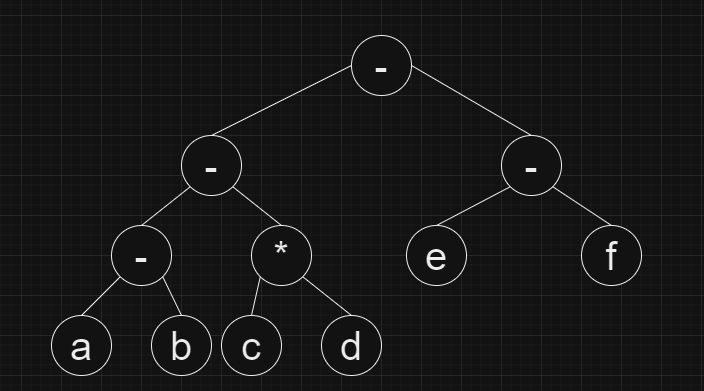
**LAB 3**

**Question 1:**

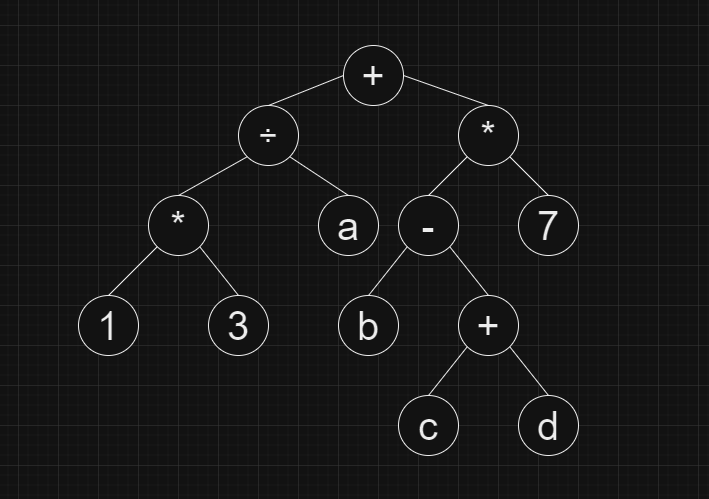
**a)** (3-a)\*(b+4)

****

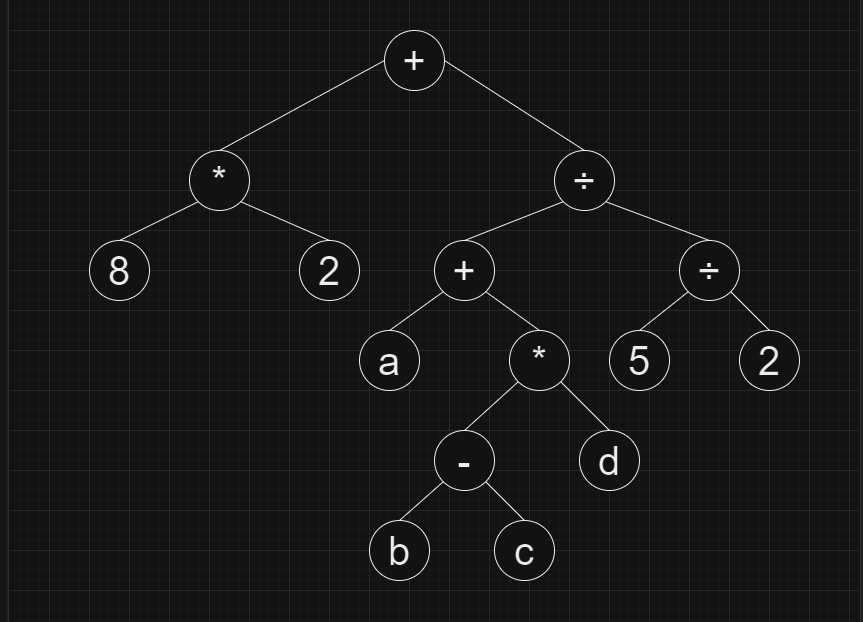
**b)** a – b – c \* d – e – f



**c)** 1 \* 3 a + (b – c + d) \* 7



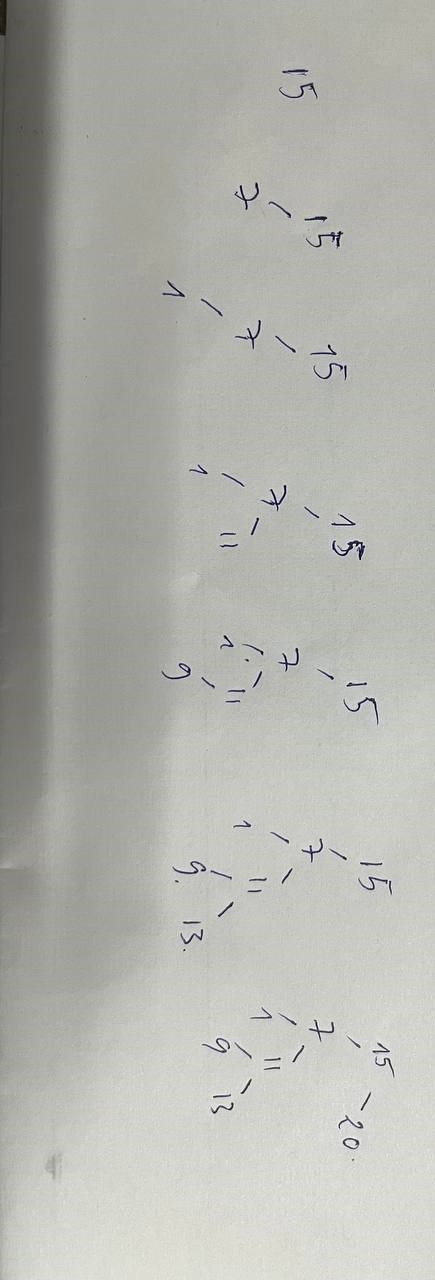
**d)** (8 \* 2) + (a + (b – c) \* d) (5 2)

****

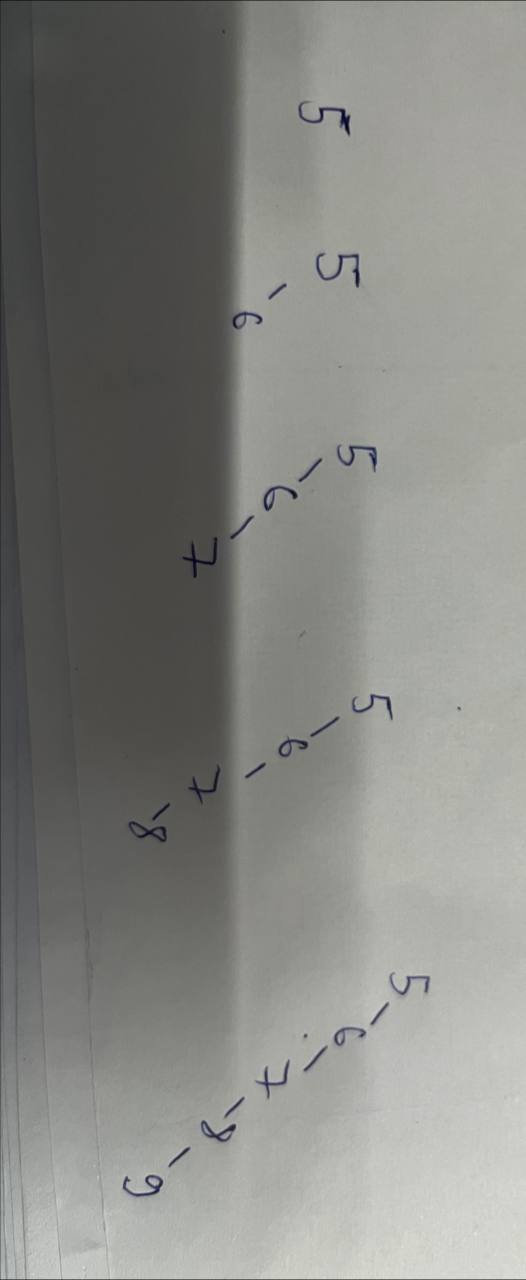
**🡪 a)** and **b)** are the complete trees because every levels, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible.

**Question 2:**

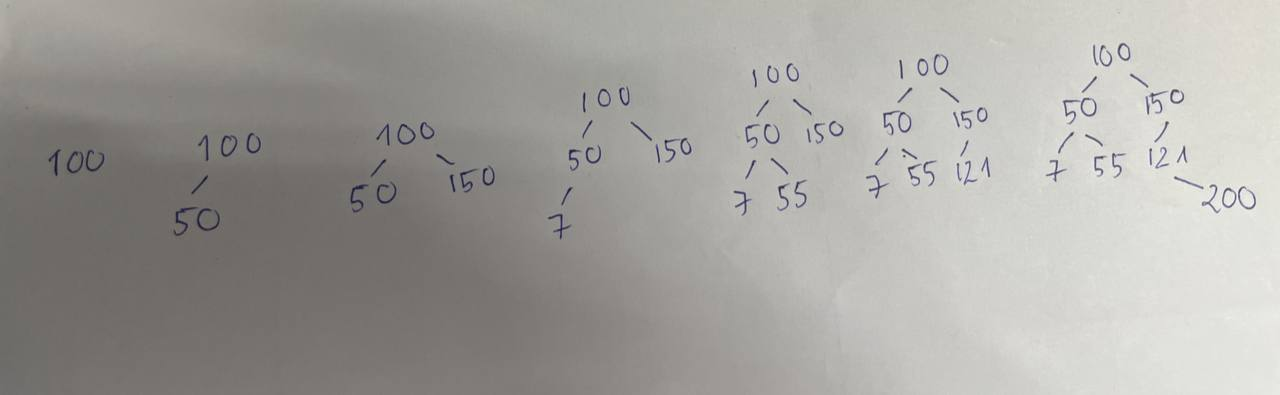
**a)** 15, 7, 1, 11, 9, 13, 20



**b)** 5, 6, 7, 8, 9



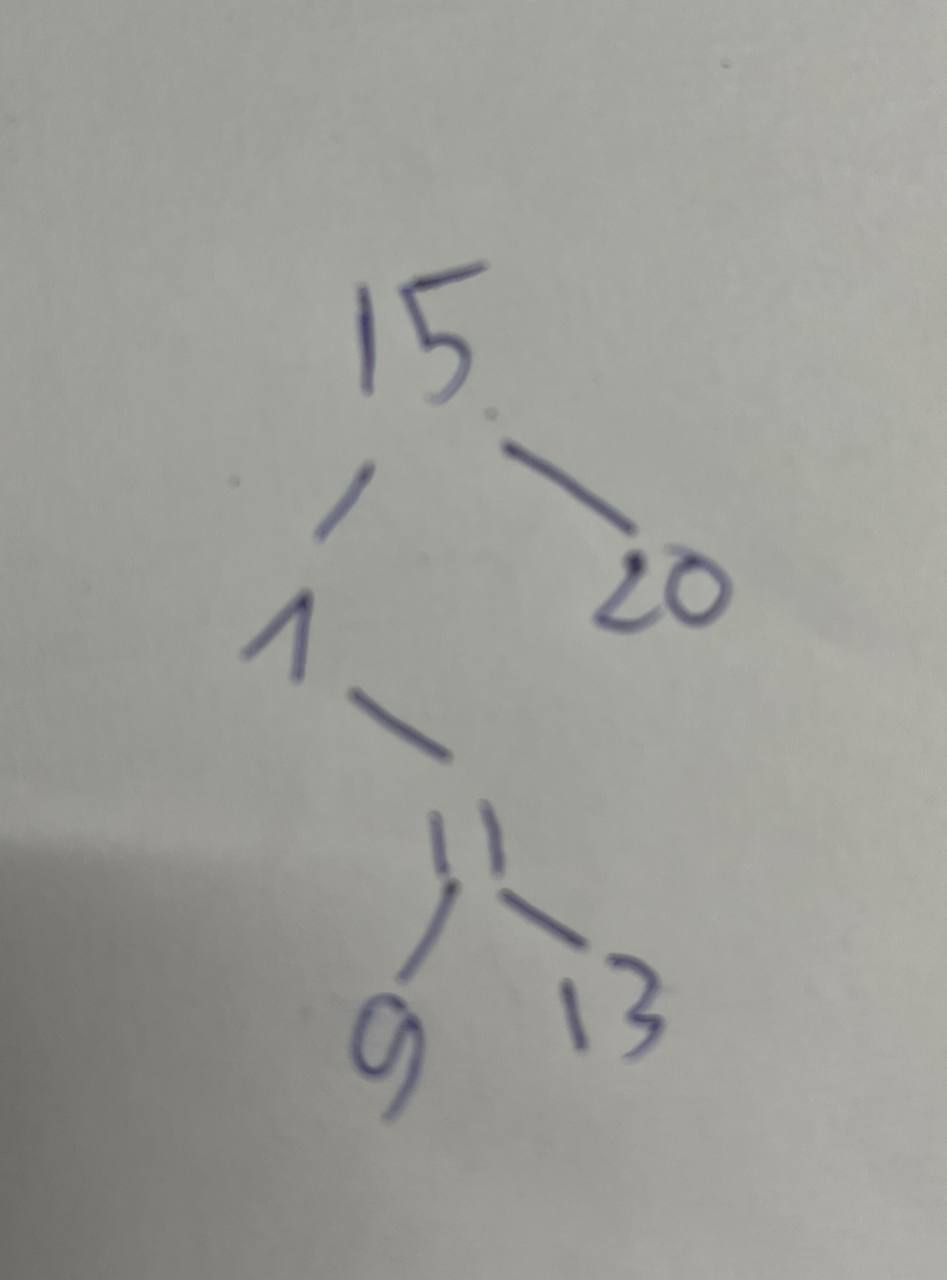
**c)** 100, 50, 150, 7, 55, 121, 200



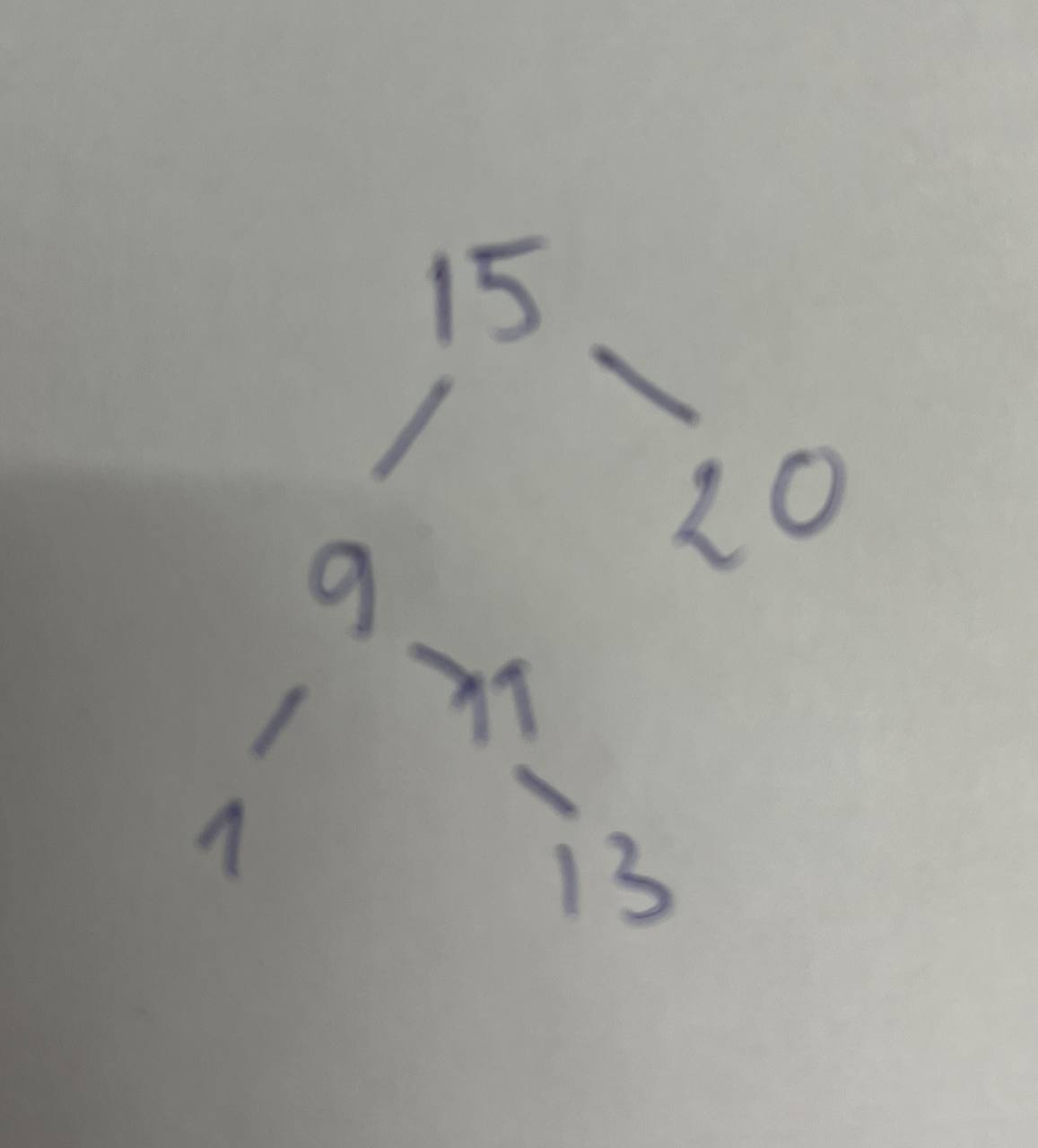
**Let’s remove key (7) in all trees. Therefore the final states are:**

**a)**

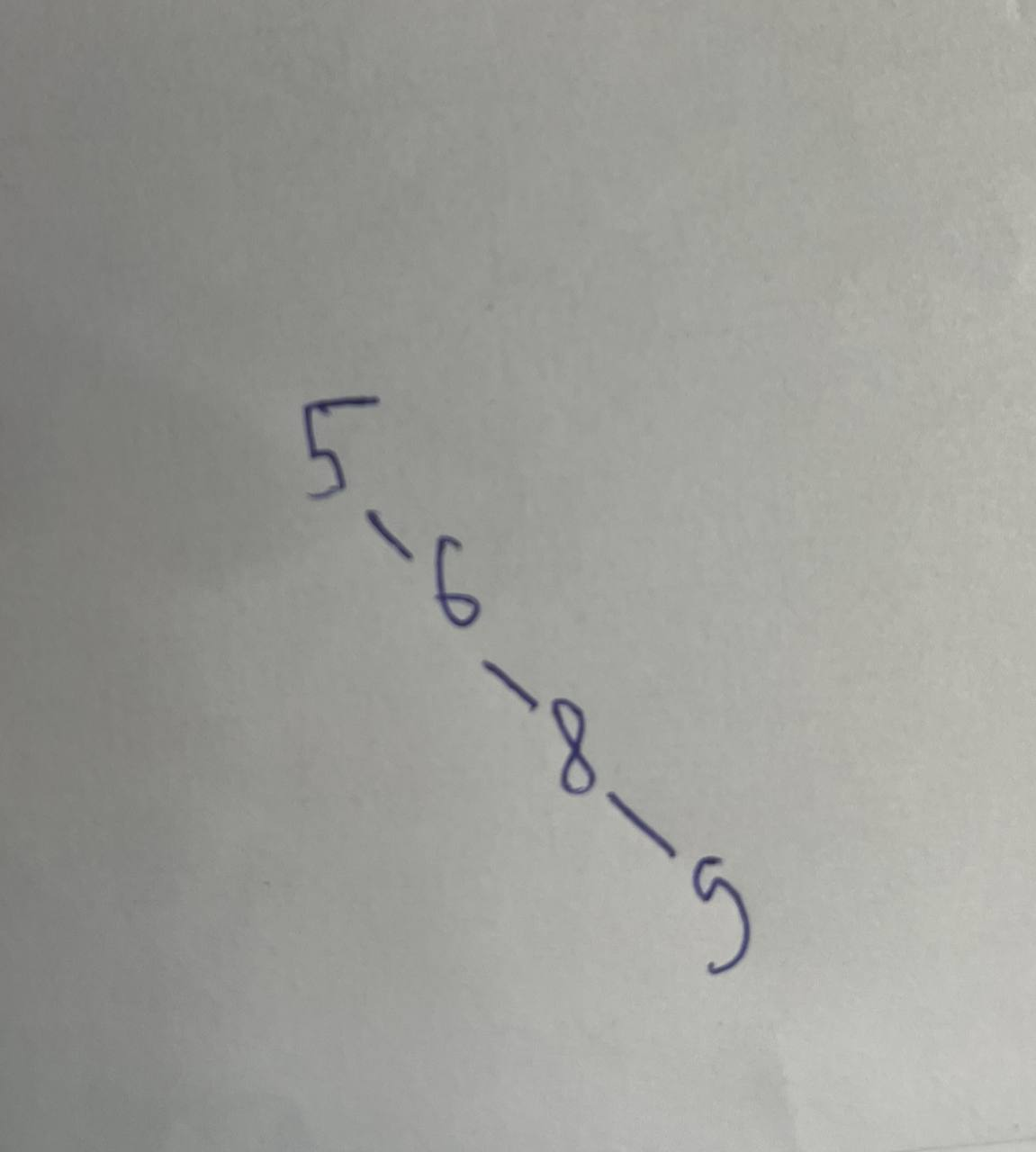
**Replace with the largest node on left subtree**



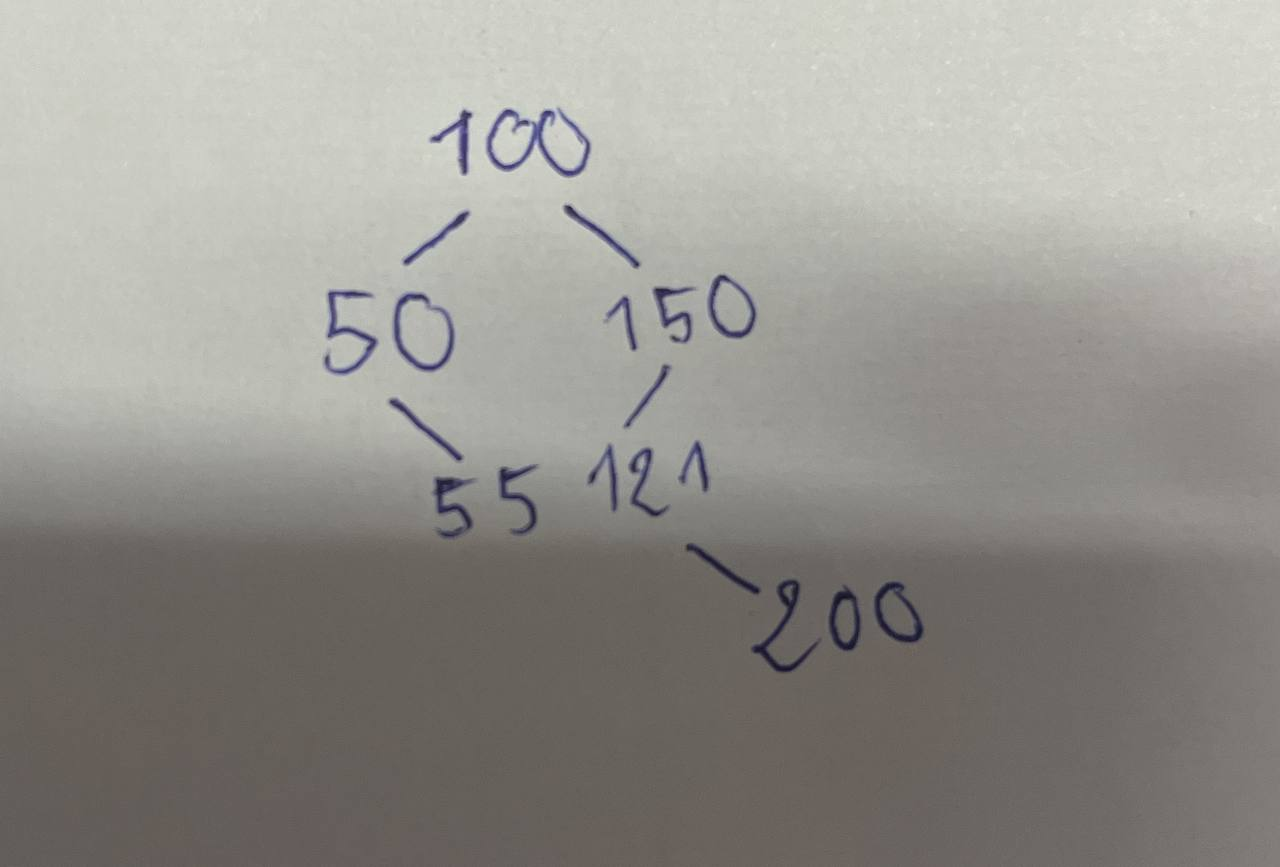
**Replace with the smallest on right subtree**



**b)**



**c)**



**Question 3:**

*treeNode*\* recursiveInsert(*treeNode*\* *subroot*, *treeNode*\* *newNode*) {

    if (*subroot* == NULL) {

*subroot* = *newNode*;

    }

    else if(*newNode* -> data <= *subroot* -> data) {

*subroot* -> left = recursiveInsert(*subroot* -> left, *newNode*);

    }

    else {

*subroot* -> right = recursiveInsert(*subroot* -> right, *newNode*);

    }

    return *subroot*;

}

**Running test:**

****

**Question 4:**

int main() {

*treeNode*\* root = **new** *treeNode*;

    root -> data = 23;

    int arr[] = {18,44,12,20,19,22,35,52};

    for (int i =0; i<sizeof(arr)/sizeof(arr[0]); i++) {

*treeNode*\* newNode = **new** *treeNode*;

        newNode -> data = arr[i];

        root = recursiveInsert(root,newNode);

    }

    traverseLNR(root,0);

}

**Question 5:**

void traverseLNR(*treeNode*\* *subroot*, int *level*) {

    if(*subroot*!=NULL) {

        int d = *level*;

        while(d>0) {

            cout<<"-";

            d = d - 1;

        }

        cout<<*subroot*->data<<endl;

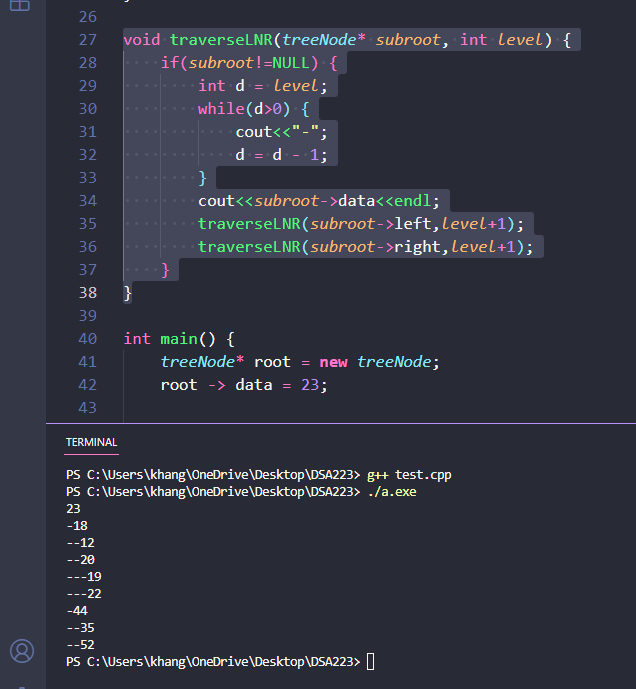
        traverseLNR(*subroot*->left,*level*+1);

        traverseLNR(*subroot*->right,*level*+1);

    }

}

**Running test for Question 5 and Question 4:**

****

**Question 6:**

void printPath(*treeNode*\* *subroot*, int *searchedData*) {

    if(*subroot* == NULL) {

        cout<<"Cannot find the data !";

    }

    else if(*searchedData* < *subroot* -> data) {

        cout<<*subroot* -> data<<" ";

        printPath(*subroot*-> left, *searchedData*);

    }

    else if(*searchedData* > *subroot* -> data) {

        cout<<*subroot* -> data<<" ";

        printPath(*subroot* -> right, *searchedData*);

    }

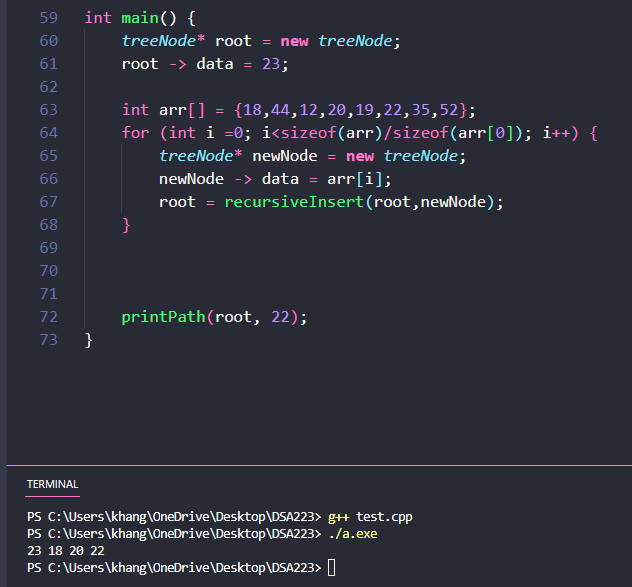
    else {

        cout<<*subroot*->data<<endl;

    }

}

**Testing result:**

****

**Question 7:**

void printLeavesIncreasingOrder(*treeNode*\* *root*) {

    if(*root*!=NULL) {

        if(*root* -> left == NULL && *root* -> right == NULL) {

            cout<<*root* -> data<<" ";

        }

        else {

            printLeavesIncreasingOrder(*root* -> left);

            printLeavesIncreasingOrder(*root*->right);

        }

    }

}

**Testing result:**

****

**Question 8:**

void printLeavesDecreasingOrder(*treeNode*\* *root*) {

    if(*root*!=NULL) {

        if(*root* -> left == NULL && *root* -> right == NULL) {

            cout<<*root* -> data<<" ";

        }

        else {

            printLeavesDecreasingOrder(*root* ->right);

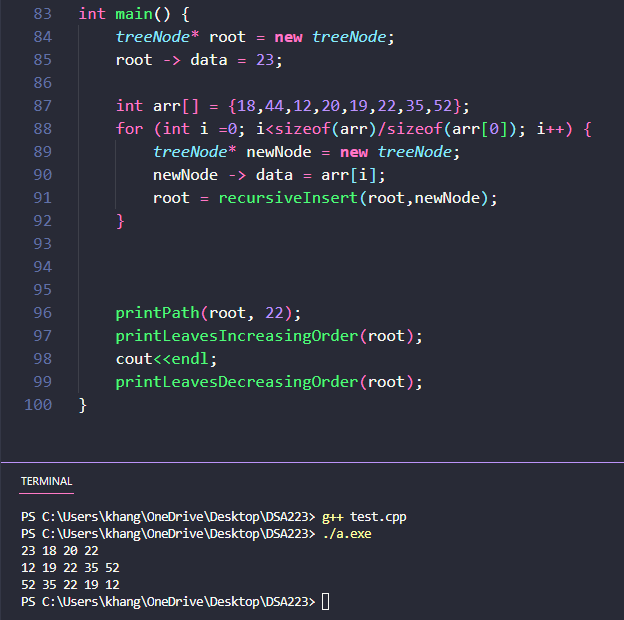
            printLeavesDecreasingOrder(*root*->left);

        }

    }

}

**Testing result:**

****

**Question 9:**

int height(*treeNode*\* *node*) {

    if(*node* == NULL) {

        return 0;

    }

    int left = 1+height(*node* -> left);

    int right = 1+height(*node* -> right);

    return (left<right)?right:left;

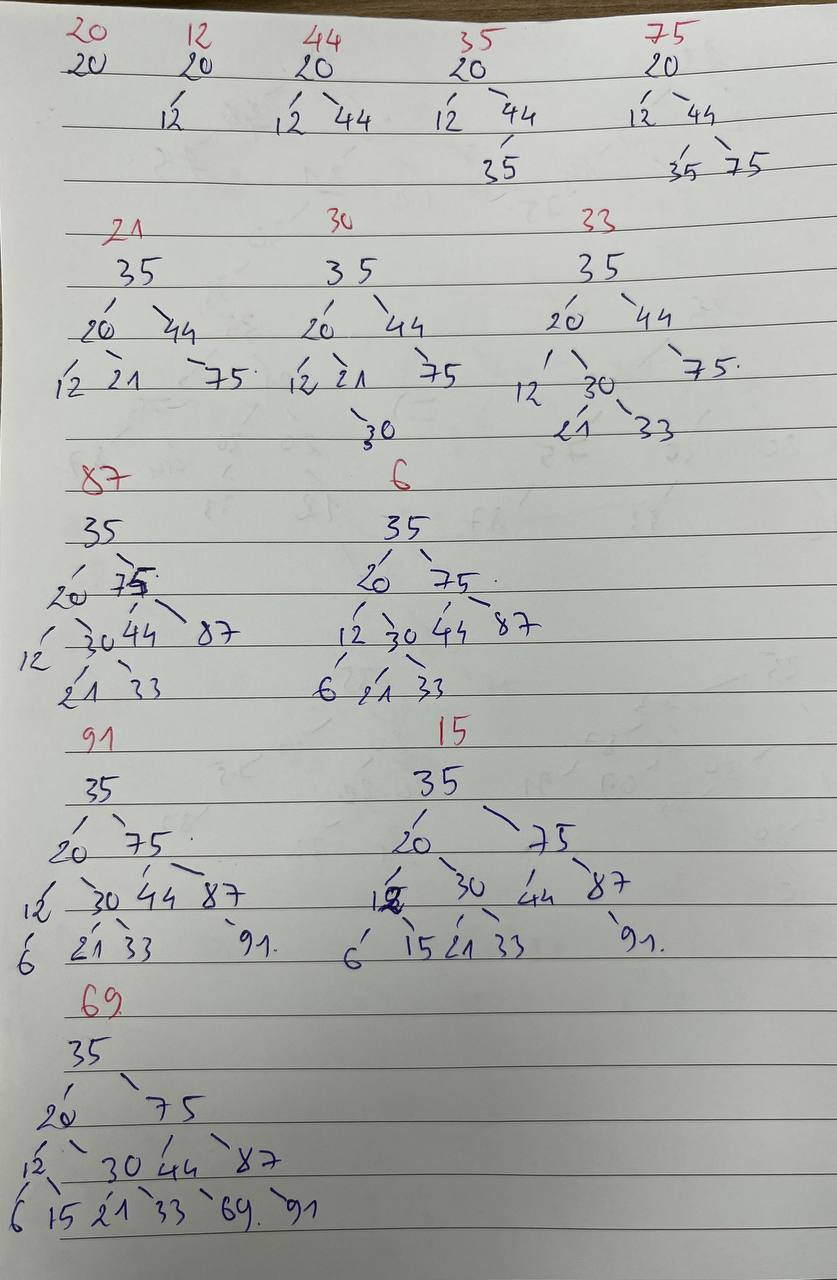
}

**Testing result:**

****

**Question 10:**

The states of tree after each inserting elements are written below



**Question 11:**

The states of tree after each deleting elements are written below.

