```cpp
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<time.h>
#include<conio.h>
#include<Windows.h>
#include<WindowsX.h>

#define LED_PIN 13;
#define BinTree struct binTree{};

//Today is Binary Tree

struct data{
    int angka;

    data *left;
    data *right; //recursive!!!
}*root = NULL;

void insertNode(data **temp, int angka){
    if(*temp==NULL){
        *temp = (data *) malloc(sizeof(data));
        (*temp)->angka = angka;
        (*temp)->left = NULL;
        (*temp)->right = NULL;
    }
    else if(angka > (*temp)->angka) insertNode(&(*temp)->right, angka);
    else if(angka < (*temp)->angka) insertNode(&(*temp)->left, angka);

}

data **getAnakKiriPalingKanan(data**temp){
    if((*temp)->right!=NULL){
        getAnakKiriPalingKanan(&(*temp)->right);
    }
    else{
        return temp;
    }
} //lecturer's


void popPengganti(data **temp){
    if((*temp)->left==NULL && (*temp)->right==NULL){
        //if leaf
        free(*temp);
        *temp=NULL;
    }
    else if((*temp)->left!=NULL && (*temp)->right==NULL){
        //if there is left child, there is no right child, right filled!
        data *temp2 = *temp;
        *temp = (*temp)->left;
        free(temp2);
    }
    else if((*temp)->right!=NULL && (*temp)->left==NULL){
        //if there is no left child, there is right child, left filled!
        data *temp2 = *temp;
        *temp = (*temp)->right;
        free(temp2);
    }
    else if((*temp)->right!=NULL && (*temp)->left!=NULL){
        //if there is left child, there is right child, both filled!
        data **curr = getAnakKiriPalingKanan(&(*temp)->left);
        (*temp)->angka = (*curr)->angka;
        popPengganti(curr); //get into this node's family
    }
} //lecturer's


void pop(data **curr,int angka){
    if((*curr)->angka == angka){
        popPengganti(curr);
    }
    else{
        if(angka>(*curr)->angka)pop(&(*curr)->right,angka);
        else if(angka<(*curr)->angka)pop(&(*curr)->left,angka);
    }
} //lecturer's

void eraseNode(data **temp, int angka){ //pop(&root, 10);
    /* Conditions
    if leaf
    if 1 child
    if 2 children
    */
    if(*temp==NULL){

    }
}

void preOrder(data **temp){
    if(*temp!=NULL){
        printf("%d ",(*temp)->angka);
        preOrder(&(*temp)->left);
        preOrder(&(*temp)->right);
    }
}

void inOrder(data **temp){
    if(*temp!=NULL){
        inOrder(&(*temp)->left);
        printf("%d ",(*temp)->angka);
        inOrder(&(*temp)->right);
    }
}

void postOrder(data **temp){
    if(*temp!=NULL){
        postOrder(&(*temp)->left);
        postOrder(&(*temp)->right);
```

```cpp
            printf("%d ",(*temp)->angka);
        }
}

void printDatas(int mode){
    data **nd = &root;

    if(mode == 0){
        /*printf("%d", (*nd)->angka);
        *nd = (*nd)->left;
        if((*nd) != NULL){
            printDatas(0);
        }*/
    } else if(mode == 1){

    } else if(mode == 2){

    }
}

int main(){
    int select = 0, kounter = 0;

    ////hmst;
    //printf("Press Enter to clrscr in Visual!");
    //getchar();

    ////printf("\033[0J");
    ////printf("%c", 12);
    //system("cls"); //bad idea!
    //system("color fc"); // here's why
    //printf("Please do not use system();!\n dangerous!");
    //getchar();
    //system("color");

    /*insertNode(&root,13);
    insertNode(&root,15);
    insertNode(&root,10);*/

    insertNode(&root,15);
    insertNode(&root,7);
    insertNode(&root,9);
    insertNode(&root,8);

    pop(&root,9);
    printf("Pre-order: ");
    //printDatas(0);
    preOrder(&root);
    printf("\n");

    printf("In-order: ");
    //printDatas(1);
    inOrder(&root);
    printf("\n");

    printf("Post-order: ");
    //printDatas(2);
    postOrder(&root);
    printf("\n");

    getchar();
    return 0;
}

//HOMEwork!
/*
Binary Tree, football club player list

view menu -> preOrder
exit and remove all -> popALL
add -> Decide between left or right of previous trees. I mean if tree is not NULL
*/
```