BST 5

```
//deletion
#include <stdio.h>
#include <stdlib.h>
int max(int a,int b)
{
	if(a>=b)
		return a;
	return b;
}
struct node
{
	int data;
	struct node* left;
	struct node* right;
};
struct node* insert(struct node* root,int x)
{
	if(root == NULL)
	{
		struct node* temp = (node*)malloc(sizeof(struct node));
		temp->data = x;
		temp->left = temp->right = NULL;
		return temp;
	}
	else if(x <= root->data)
		root->left = insert(root->left,x);
	else
		root->right = insert(root->right,x);
	return root;
}
struct node* findmin(struct node* root)
{
	if(root->left == NULL)
		return root;
	return findmin(root->left);
}
struct node* del(struct node* root,int x)
{
	if(root == NULL) return root;
	else if(x < root->data) root->left = del(root->left,x);
	else if(x > root->data) root->right = del(root->right,x);
	else
	{
		//case 1
		if(root->left == NULL && root->right == NULL)
		{
			free(root);
			return NULL;
```

```
                }
                //case 2
                else if(root->left == NULL)
                {
                        struct node* temp = root;
                        root = root->right;
                        free(temp);
                }
                else if(root->right == NULL)
                {
                        struct node* temp = root;
                        root = root->left;
                        free(temp);
                }
                //case 3
                else
                {
                        struct node* temp = findmin(root->right);
                        root->data = temp->data;
                        root->right = del(root->right,temp->data);
                }
        }
        return root;
}
void inorder(struct node* root)
{
        if(root == NULL) return;
        inorder(root->left);
        printf("%d ",root->data);
        inorder(root->right);
}
int main()
{
        int x,n;
        struct node* root = NULL;
        while(1)
        {
                //printf("1 insert,2 del,3 end\n");
                scanf("%d",&n);
                switch(n)
                {
                        case 1:
                                scanf("%d",&x);
                                root = insert(root,x);
                                break;
                        case 2:
                                scanf("%d",&x);
                                root = del(root,x);
                                break;
                        case 3:
                                inorder(root);
                                break;
```

```
                case 4:
                        return 0;
                }
        }

}
```