The Primary / Foundation BST Code

Develop and Test the BST – create, delete, search, print. This will be the core logic. Now customize it for this problem domain. Core code is as follows.

//Writer: Rajesh D. Nagawade

// gets a node, inserts, deletes a node and traverses the tree

```
 # include <stdio.h>

# include <stdlib.h>

typedef struct Node
{
    int value;
    struct Node* left;
    struct Node* right;
} ;

Node* root = NULL;

int A[10] = {20,10,30,25,35,5,15,7,13,27};

Node* getnode()
{
    static int i = 0;

    Node* newNode = (Node*)malloc(sizeof(Node));
    if (newNode == NULL) {
        printf("Error allocating memory\n");
        exit(1);
    }

    //printf("enter the value of node : ");

    //scanf("%d",&(newNode->value));

    newNode -> value = A[i++];

    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

void insert(Node* n) {
```

```
    Node* p = root;
    Node* q;
    while (p != NULL)
    {
        q = p;
        if (n->value < p->value) {
            p = p->left;
        } else {
            p = p->right;
        }
    }

    if (n->value < q->value) {

        printf("inserting %d on left of %d",n->value, q->value);
        q->left = n;
        printf("\n");
    } else {
        printf("inserting %d on right of %d",n->value, q->value);
        q->right = n;
        printf("\n");
    }
}

void Del(int no)
{

    Node* p=root;
    Node* q=root;
    while(no!=p->value)
    {
        if(no<p->value)
        {
            q=p;
            p=p->left;
        }
        else
        {
            q=p;
            p=p->right;
        }
    } // here p reaches the node to delete

    if(p->left==NULL && p->right==NULL)// delete if it has no child
    {
        printf("deleting %d \n", p->value);

        if((q->left) && (q->left->value==no))
            q->left=NULL;
        else
```

```
            q->right=NULL;
    }
    else // don't delete
            printf("Can't delete %d - has some children \n", p->value);

    free(p);
}

int main()
{
    Node* k;
    root = getnode();

    for (int i = 0; i <=8; i++) {
        k = getnode();
        insert(k);
    }

    Del(25);

    printf("\n");
    return 0;
}
```
