WEEK13

BINARY SEARCH TREE:

- **Display minimum/maximum**
- **Display number of nodes**

CODE:

```
#include <stdio.h>
#include <stdlib.h>
int nnodes = 0;
struct node
{
	int info;
	struct node *rlink;
	struct node *llink;
};
typedef struct node *NODE;
NODE getnode()
{
	NODE x;
	x = (NODE)malloc(sizeof(struct node));
	if (x == NULL)
	{
		printf("mem full\n");
		exit(0);
	}
	return x;
}
void freenode(NODE x)
{
	free(x);
}
NODE insert(NODE root, int item)
{
	NODE temp, cur, prev;
	temp = getnode();
	temp->rlink = NULL;
	temp->llink = NULL;
	temp->info = item;
	nnodes = nnodes + 1;
```

```
        if (root == NULL)
                return temp;
        prev = NULL;
        cur = root;
        while (cur != NULL)
        {
                prev = cur;
                cur = (item < cur->info) ? cur->llink : cur->rlink;
        }
        if (item < prev->info)
                prev->llink = temp;
        else
                prev->rlink = temp;
        return root;
}
void display(NODE root, int i)
{
        int j;
        if (root != NULL)
        {
                display(root->rlink, i + 1);
                for (j = 0; j < i; j++)
                        printf("    ");
                printf("%d\n", root->info);
                display(root->llink, i + 1);
        }
}
void dispmax_min(NODE root)
{
        NODE temp, root2;
        root2 = root;
        temp = root->llink;
        while (temp != NULL)
        {
                temp = temp->llink;
                root = root->llink;
        }
        printf("Minimum: %d\n", root->info);
        temp = root2->rlink;
        while (temp != NULL)
        {
                temp = temp->rlink;
                root2 = root2->rlink;
```

```
        }
        printf("Minimum: %d\n", root2->info);
}
NODE delete (NODE root, int item)
{
        nnodes = nnodes - 1;
        NODE cur, parent, q, suc;
        if (root == NULL)
        {
                printf("empty\n");
                return root;
        }
        parent = NULL;
        cur = root;
        while (cur != NULL && item != cur->info)
        {
                parent = cur;
                cur = (item < cur->info) ? cur->llink : cur->rlink;
        }
        if (cur == NULL)
        {
                printf("Not found\n");
                return root;
        }
        if (cur->llink == NULL)
                q = cur->rlink;
        else if (cur->rlink == NULL)
                q = cur->llink;
        else
        {
                suc = cur->rlink;
                while (suc->llink != NULL)
                        suc = suc->llink;
                suc->llink = cur->llink;
                q = cur->rlink;
        }
        if (parent == NULL)
                return q;
        if (cur == parent->llink)
                parent->llink = q;
        else
                parent->rlink = q;
        freenode(cur);
```

```
        return root;
}

void preorder(NODE root)
{
        if (root != NULL)
        {
                printf("%d\n", root->info);
                preorder(root->llink);
                preorder(root->rlink);
        }
}
void postorder(NODE root)
{
        if (root != NULL)
        {

                postorder(root->llink);
                postorder(root->rlink);
                printf("%d\n", root->info);
        }
}
void inorder(NODE root)
{
        if (root != NULL)
        {

                inorder(root->llink);
                printf("%d\n", root->info);
                inorder(root->rlink);
        }
}
void main()
{
        int item, choice;
        NODE root = NULL;

        for (;;)
        {
                printf("\n1.Insert\n2.Display\n3.Pre-order\n4.Post-order\n5.In-order\n6.Delete\n7.Display
max/min\n8.display_num_nodes\n9.Exit\n");
                printf("Enter the choice\n");
                scanf("%d", &choice);
```

```
            switch (choice)
            {
            case 1:
                  printf("Enter the item\n");
                  scanf("%d", &item);
                  root = insert(root, item);
                  break;
            case 2:
                  printf("Binary search tree:\n");
                  display(root, 0);
                  break;
            case 3:
                  preorder(root);
                  break;
            case 4:
                  postorder(root);
                  break;
            case 5:
                  inorder(root);
                  break;
            case 6:
                  printf("Enter the item\n");
                  scanf("%d", &item);
                  root = delete (root, item);
                  break;
            case 7:
                  dispmax_min(root);
                  break;
            case 8:
                  printf("Number of nodes: %d\n", nnodes);
                  break;
            default:
                  exit(0);
                  break;
            }
      }
}
```

OUTPUT:

```
Binary search tree:
  25
10
    8
  5
    2

1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Display max/min
8.display_num_nodes
9.Exit
Enter the choice
7
Minimum: 2
Minimum: 25
```

```
1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Display max/min
8.display_num_nodes
9.Exit
Enter the choice
8
Number of nodes: 5

1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Display max/min
8.display_num_nodes
9.Exit
Enter the choice
9

C:\Users\misaf\Desktop\DS LAB\week13>
```