1BM19CS090 RAHUL

Lab Program 10: BINARY SEARCH TREE.

Code:

```
#include<stdio.h>
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("Memory 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;
    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 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(NODE root->rlink);
        printf("%d\n", root->info);
    }
}
```

```
void inorder(NODE root)
{
    if(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. Exit\n");
        printf("Enter choice : ");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1: printf("Enter the Item : ");
                    scanf("%d", &item);
                    root = insert(root, item);
                    break;
            case 2: display(root, 0);
                    break;
            case 3: preorder(root);
                    break;
            case 4: Inorder(root);
                    break;
            case 5: Postorder(root);
                    break;
            default: exit(0);
                    break;
        }
    }
}
```

BinarySearchTree.c

Saved

```c
#include<stdio.h>
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("Memory 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;
    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 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.In-Order\n5.Post-Order\n6.Exit\n");
    printf("Enter Choice:");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
            printf("Enter Item:");
        scanf("%d",&item);
        root=insert(root,item);
        break;
        case 2:
            display(root,0);
        break;
        case 3:
            preorder(root);
        break;
        case 4:
            inorder(root);
        break;
        case 5:
            postorder(root);
        break;
        default:
            exit(0);
        break;
   }
    }
}
```

```
Terminal

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:100

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:20

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:10

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:30

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:200

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:150

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:1
Enter Item:300

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:3
100
20
10
30
200
150
300

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:4
10
20
30
100
150
200
300

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:5
10
30
20
150
300
200
100

1.Insert
2.Display
3.Pre-Order
4.In-Order
5.Post-Order
6.Exit
Enter Choice:6

Process finished.
```