Slips answers for BST 🤷‍♂️🤷‍♀️😱

Q.1)Implement a Binary search tree (BST) library (btree.h) with operations – create, insert, inorder. Write a menu driven program that performs the above operations. [15]

→ .C file

```
#include<stdio.h>
#include<stdlib.h>
#include"1btree.h"
int main(){
   struct node* root=NULL;
   int choice,data;
   struct node* result;

   do{
      displaymenu();
      scanf("%d",&choice);

      switch(choice){
         case 1:
         printf("Enter the value to insert : ");
         scanf("%d",&data);
         root=insert(root,data);
         break;

         case 2:
         printf("Inorder traversal :");
         inorder(root);
         printf("\n");
         break;

      }
   } while(choice!=2);
   return 0;
}
```

.H file

```
struct node{
   int value;
   struct node* left;
   struct node* right;
};

//function to create a new node //
struct node* createnode(int data){
   struct node* newnode=(struct node*)malloc(sizeof(struct node));
   newnode->value=data;
   newnode->left=newnode->right=NULL;
   return newnode;
}

struct node* insert(struct node* root,int data){
   if(root==NULL){
      return createnode(data);
   }
   if(root->value<data){
      root->right=insert(root->right,data);
   }
   if(root->value>data){
      root->left=insert(root->left,data);
   }
   return root;
}
void inorder(struct node* root){
   if(root!=NULL){
      inorder(root->left);
      printf("%d ",root->value);
      inorder(root->right);
   }
}
void displaymenu(){
   printf("\n 1. insert  a node : ");
   printf("\n 2. inorder : ");
   printf("\n 3. Exit ");
   printf("\n\n Enter the choice : ");
}
```

OUTPUT :

```
PS E:\TREES_Slips> gcc 1btree.c
PS E:\TREES_Slips> ./a.exe

 1. insert  a node :
 2. inorder :
 3. Exit

 Enter the choice : 1
Enter the value to insert : 10

 1. insert  a node :
 2. inorder :
 3. Exit

 Enter the choice : 1
Enter the value to insert : 20

 1. insert  a node :
 2. inorder :
 3. Exit

 Enter the choice : 1
Enter the value to insert : 30

 1. insert  a node :
 2. inorder :
 3. Exit

 Enter the choice : 2
Inorder traversal :10 20 30
```

Q.2) Implement a Binary search tree (BST) library (btree.h) with operations – create, search, preorder. Write a menu driven program that performs the above operations. [15]

→

. C file

```
#include<stdio.h>
#include<stdlib.h>
#include"2btree.h"
int main(){
   struct node* root=NULL;
   int choice,data;
   struct node* result;

   do{
      displaymenu();
      scanf("%d",&choice);

      switch(choice){
         case 1:
         printf("Enter the value to insert : ");
         scanf("%d",&data);
         root=insert(root,data);
         break;

         case 2:
         printf("enter the value to be search :");
         scanf("%d",&data);
         result=search(root,data);
         if(result!=NULL){
            printf("value %d found in the BST\n",data);
         }
         else{
            printf("value %d not found in the BST\n",data);
         }
         break;
```

```
            case 3:
            printf("preorder traversal ");
            preorder(root);
            printf("\n");
            break;

        }
    } while(choice!=3);
    return 0;
}
```

.H file

```
struct node{
    int value;
    struct node* left;
    struct node* right;
};

//function to create a new node //
struct node* createnode(int data){
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->value=data;
    newnode->left=newnode->right=NULL;
    return newnode;
}

struct node* insert(struct node* root,int data){
    if(root==NULL){
        return createnode(data);
    }
    if(root->value<data){
        root->right=insert(root->right,data);
    }
    if(root->value>data){
        root->left=insert(root->left,data);
    }
    return root;
}

void preorder(struct node* root){
```

```
    if(root!=NULL){
        printf("%d ",root->value);
        preorder(root->left);
        preorder(root->right);
    }
}



struct node* search(struct node* root,int data){
    if(root==NULL || root->value==data){
        return root;
    }
    if(root->value>data){
        return search(root->left,data);
    }
    else{
        return search(root->right,data);
    }
}

void displaymenu(){
    printf("\n 1. insert  a node : ");
    printf("\n 2. search for a value : ");
    printf("\n 3. preorder : ");
    printf("\n 4. Exit ");
    printf("\n\n Enter the choice : ");
}
```

Q3) Implement a Binary search tree (BST) library (btree.h) with operations – create, search, preorder. Write a menu driven program that performs the above operations. [15]

—> Follow que 2

Q.4) Implement a Binary search tree (BST) library (btree.h) with operations – create, insert, postorder. Write a menu driven program that performs the above operations. [15]

—>

. C file

```
#include<stdio.h>
#include<stdlib.h>
#include"4btree.h"
int main(){
   struct node* root=NULL;
   int choice,data;
   struct node* result;

   do{
      displaymenu();
      scanf("%d",&choice);

      switch(choice){
         case 1:
         printf("Enter the value to insert : ");
         scanf("%d",&data);
         root=insert(root,data);
         break;

         case 2:
         printf("enter the value to be search :");
         scanf("%d",&data);
         result=search(root,data);
         if(result!=NULL){
            printf("value %d found in the BST\n",data);
         }
         else{
            printf("value %d not found in the BST\n",data);
         }
         break;

         case 3:
         printf("postorder traversal :");
         postorder(root);
         printf("\n");
         break;
         default:
```

```
            printf("invalid choice\n");
        }
    } while(choice!=3);
    return 0;
}
```

.H file

```
struct node{
    int value;
    struct node* left;
    struct node* right;
};

//function to create a new node //
struct node* createnode(int data){
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->value=data;
    newnode->left=newnode->right=NULL;
    return newnode;
}

struct node* insert(struct node* root,int data){
    if(root==NULL){
        return createnode(data);
    }
    if(root->value<data){
        root->right=insert(root->right,data);
    }
    if(root->value>data){
        root->left=insert(root->left,data);
    }
    return root;
}

void postorder(struct node* root){
    if(root!=NULL){
        postorder(root->left);
        postorder(root->right);
        printf("%d ",root->value);
    }
}
```

```
struct node* search(struct node* root,int data){
   if(root==NULL || root->value==data){
      return root;
   }
   if(root->value>data){
      return search(root->left,data);
   }
   else{
      return search(root->right,data);
   }
}

void displaymenu(){
   printf("\n 1. insert  a node : ");
   printf("\n 2. search for a value : ");
   printf("\n 3. postorder : ");
   printf("\n 4. Exit ");
   printf("\n\n Enter the choice : ");
}
```

Q.5) Implement a Binary search tree (BST) library (btree.h) with operations – create, preorder and postorder. Write a menu driven program that performs the above operations. [15]

→

.C file

```
#include<stdio.h>
#include<stdlib.h>
#include"5btree.h"
int main(){
   struct node* root=NULL;
   int choice,data;
   struct node* result;

   do{
```

```
        displaymenu();
        scanf("%d",&choice);

        switch(choice){
            case 1:
            printf("Enter the value to insert : ");
            scanf("%d",&data);
            root=insert(root,data);
            break;

            case 2:
            printf("preorder traversal ");
            preorder(root);
            printf("\n");
            break;

            case 3:
            printf("postorder traversal :");
            postorder(root);
            printf("\n");
            break;
            default:
            printf("invalid choice\n");
        }
    } while(choice!=3);
    return 0;
}
```

.H file

```
struct node{
    int value;
    struct node* left;
    struct node* right;
};

//function to create a new node //
struct node* createnode(int data){
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->value=data;
    newnode->left=newnode->right=NULL;
    return newnode;
```

```
}

struct node* insert(struct node* root,int data){
   if(root==NULL){
      return createnode(data);
   }
   if(root->value<data){
      root->right=insert(root->right,data);
   }
   if(root->value>data){
      root->left=insert(root->left,data);
   }
   return root;
}


void preorder(struct node* root){
   if(root!=NULL){
      printf("%d ",root->value);
      preorder(root->left);
      preorder(root->right);
   }
}

void postorder(struct node* root){
   if(root!=NULL){
      postorder(root->left);
      postorder(root->right);
      printf("%d ",root->value);
   }
}

void displaymenu(){
   printf("\n 1. insert  a node : ");
   printf("\n 2. preorder : ");
   printf("\n 3. postorder : ");
   printf("\n 4. Exit ");
   printf("\n\n Enter the choice : ");
}
```

Q.6) Implement a Binary search tree (BST) library (btree.h) with operations – create, preorder .Write a menu driven program that performs the above operations. [15]

→ Follow que 2

Q.7) C program to implement BST to perform following operations on BST- a) Create b) Counting leaf nodes

→

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *left;
   struct node *right;
};
struct node *createnode(int data){
   struct node *newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=data;
   newnode->left=NULL;
   newnode->right=NULL;
   return newnode;
}

struct node *insert(struct node *root,int data){
   if(root==NULL){
      return createnode(data);
   }
   if(data<root->data){
      root->left=insert(root->left,data);
   }
   if(data>root->data){
      root->right=insert(root->right,data);
```

```
    }
    return root;
}

int countleafnode(struct node *root){
    if(root==NULL){
        return 0;
    }
    if(root->left==NULL && root->right==NULL){
        return 1;
    }
    return countleafnode(root->left) + countleafnode(root->right);
}

void preorder(struct node *root){
    if(root!=NULL){
        printf("%d ",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

int main(){
    struct node *root=NULL;
    root=insert(root,50);
    insert(root,30);
    insert(root,70);
    insert(root,20);
    insert(root,40);
    insert(root,60);
    insert(root,80);

    printf("preorder traversal :");
    preorder(root);
    printf("\n");

    printf("Number of the leaf nodes : %d\n",countleafnode(root));
    return 0;


}
```

```
PS E:\TREES_Slips> gcc 7.c
PS E:\TREES_Slips> ./a.exe
preorder traversal :50  30  20  40  70  60  80
Number of the leaf nodes : 4
```

Q.8) Implement a Binary search tree (BST) library (btree.h) with operations – create, inorder. Write a menu driven program that performs the above operations

→ Follow que 1

Q.9) Implement a Binary search tree (BST) library (btree.h) with operations – create, preorder. Write a menu driven program that performs the above operations.

—> Follow que 2

Q10) Write a C program which uses the Binary search tree library and implements the following function with recursion: int compare(T1, T2) – compares two binary search trees and returns 1 if they are equal and 0 otherwise.

—>

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *left;
   struct node *right;
};
struct node *createnode(int data){
   struct node *newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=data;
   newnode->left=NULL;
   newnode->right=NULL;
```

```
    return newnode;
}

struct node *insert(struct node *root,int data){
    if(root==NULL){
        return createnode(data);
    }
    if(data<root->data){
        root->left=insert(root->left,data);
    }
    if(data>root->data){
        root->right=insert(root->right,data);
    }
    return root;
}

int areEqual(struct node *root1,struct node *root2){
    if(root1==NULL && root2==NULL){
        return 1;
    }
    if(root1==NULL || root2==NULL){
        return 0;
    }
    if(root1->data!=root2->data){
        return 0;
    }
    return areEqual(root1->left,root2->left) && areEqual(root1->right,root2->right);
}



void preorder(struct node *root){
    if(root!=NULL){
        printf("%d ",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

int main(){
    struct node *root1=NULL;
```

```
    root1=insert(root1,50);
    insert(root1,30);
    insert(root1,70);
    insert(root1,20);
    insert(root1,40);
    insert(root1,60);
    insert(root1,180);

    struct node *root2=NULL;
    root2=insert(root2,50);
    insert(root2,30);
    insert(root2,70);
    insert(root2,20);
    insert(root2,40);
    insert(root2,60);
    insert(root2,80);

    printf("\n");
    printf("preorder traversal :");
    preorder(root1);
    printf("\n");

    printf("preorder traversal :");
    preorder(root2);
    printf("\n");

    if(areEqual(root1,root2)){
        printf("Two trees are equal \n");
    }
    else{
        printf("Two trees are not equal\n");
    }
    return 0;
}
```

OUTPUT :

```
PS E:\TREES_Slips> gcc 10.c
PS E:\TREES_Slips> ./a.exe
preorder traversal :50 30 20 40 70 60 180
preorder traversal :50 30 20 40 70 60 80
Two trees are not equal


PS E:\TREES_Slips> ./a.exe
preorder traversal :50 30 20 40 70 60 80
preorder traversal :50 30 20 40 70 60 80
Two trees are equal
```

Q.11) Write a program which uses a binary search tree library and counts the total nodes in the tree. int count(T) – returns the total number of nodes from BST

—>

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *left;
   struct node *right;
};
struct node *createnode(int data){
   struct node *newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=data;
   newnode->left=NULL;
   newnode->right=NULL;
   return newnode;
}

struct node *insert(struct node *root,int data){
   if(root==NULL){
```

```
        return createnode(data);
    }
    if(data<root->data){
        root->left=insert(root->left,data);
    }
    if(data>root->data){
        root->right=insert(root->right,data);
    }
    return root;
}

int countTotalNodes(struct node *root){
    if(root==NULL){
        return 0;
    }
    return 1+countTotalNodes(root->left) + countTotalNodes(root->right);
}



void preorder(struct node *root){
    if(root!=NULL){
        printf("%d ",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

int main(){
    struct node *root=NULL;
    root=insert(root,50);
    insert(root,30);
    insert(root,70);
    insert(root,20);
    insert(root,40);
    insert(root,60);
    insert(root,80);

    printf("preorder traversal :");
    preorder(root);
    printf("\n");
```

```
    printf("Total number of the nodes : %d\n",countTotalNodes(root));
    return 0;


}
```

OUTPUT :

```
PS E:\TREES_Slips> gcc 11.c
PS E:\TREES_Slips> ./a.exe
preorder traversal :50 30 20 40 70 60 80
Total number of the nodes : 7
```

Q.12) Implement a Binary search tree (BST) library (btree.h) with operations – create, preorder. Write a menu driven program that performs the above operations

—> Follow que 2

Q.13) Write a program which uses a binary search tree library and counts the total leaf nodes in the tree. int countLeaf(T) – returns the total number of leaf nodes from BST

→ Follow que 7

Q.14) Write a program which uses a binary search tree library and counts the total leaf nodes in the tree. int countLeaf(T) – returns the total number of leaf nodes from BST

—> Follow que 7

Q.15) Implement a Binary search tree (BST) library (btree.h) with operations – create, preorder. Write a menu driven program that performs the above operations

→ Follow que 2

Q.16) Implement a Binary search tree (BST) library (btree.h) with operations – create, insert, inorder. Write a menu driven program that performs the above operations.

→ Follow que 1

Q.17) Implement a Binary search tree (BST) library (btree.h) with operations – create, insert, postorder. Write a menu driven program that performs the above operations.

→Follow que 4

Q18) C program to implement BST to perform following operations on BST-

a) Create b) Counting Total nodes

—> Follow que 11

Q.19) Implement a Binary search tree (BST) library (btree.h) with operations – create, insert, postorder. Write a menu driven program that performs the above operations.

—> Follow que 4

Q.20) Implement a Binary search tree (BST) library (btree.h) with operations – create, preorder and postorder. Write a menu driven program that performs

the above operations.

→ Follow que 5