Program 11: Binary Search Tree

Date
Page

a) construct a binary search tree
b) traverse using all three methods
c) display the elements

pseudocode: a)
```
struct Node* insert(struct Node* node,
    int key){
    if(node==NULL) return newNode(key);
    if(key < node->key)
        node->left = insert(node->left, key);
    else if(key > node->key)
        node->right = insert(node->right, key);
    return node;}
```

b)
```
void inorder(struct Node* root){
    if(root != NULL){
        inorder(root->left);
        printf("%d", root->key);
        inorder(root->right);}}

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

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

9/12/25

code:

```
#include <stdio.h>
#include <stdlib.h>

struct Node{
    int data;
    struct Node *left, *right;
};

struct Node* createNode(int value){
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}

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

void inorder(struct Node *root){
    if (root==NULL) return;
    inorder(root->left);
    printf("%d", root->data);
    inorder(root->right);
}

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

```
    if (root==NULL) return;
    printf ("%d", root->data);
    preorder(root->left);
    preorder(root->right);
}

void postorder (struct Node* root){
    if (root==NULL) return;
    postorder (root->left);
    postorder (root->right);
    printf ("%d", root->data);
}

void displayTree (struct Node *root){
    int level=0, indentSpace=5;i;
    if(root == NULL) return;

    for(i=0; i< level* indentSpace; i++)
        printf(" ");
    printf("%d \n", root->data);

    if (root->left!=NULL){
        for(i=0; i< level * indentSpace; i++)
            printf(" ");
        printf("/\n");
        printTree(root->left, level+1, indentSpace);
    }

    if (root->right !=NULL){
        for(i=0; i< level * indentSpace; i++);
            printf(" ");
        printf("\\n");
```

```
        printTree(root->right, level+1, indentSpace);
    }
}

int main() {
    struct Node *root = NULL;
    int choice, value;
    printf("1: Insert into BST\n");
    printf("2: Inorder\n");
    printf("3: Preorder\n");
    printf("4: Postorder\n");
    printf("5: Display");
    printf("6: Exit\n");
    while(1) {
        switch(choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                root = insert(root, value);
                break;

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

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

```
case 4:
    printf("postorder traversal: ");
    postorder(root);
    printf("\n");
    break;
case 5:
    display tree(root);
    break;
case 6:
    exit 0;
default:
    printf("invalid choee! Try again \n");
}
}
return 0;
}
```

Output:

```
--Binary Search Tree Menu--
1. Insert into BST
2. Inorder
3. Preorder
4. Postorder
5. Exit
Enter choice 1
Enter value 10
Enter choice 1
Enter value 30
Enter choice 1
Enter value 5
Enter choice 1
Enter value 100
enter choice: 3
preorder: 10 5 30 100
enter choice: 2
inorder: 5 10 30 100
enter choice: 4
post order: 5 100 30 10
enter choice: 5
BST Elements: 5 10 30 100
enter choice: 6
```