**/\* Write a program to implement create (insert) or traverse (display) operations**

**on binary search tree (BST) without using recursion for insert or create function. \*/**

**#include <stdio.h>**

**#include <conio.h>**

**#include <stdlib.h>**

**struct tree**

**{**

**struct tree \*left;**

**int value;**

**struct tree \*right;**

**};**

**void insert();**

**void preorder(struct tree \*node);**

**void inorder(struct tree \*node);**

**void postorder(struct tree \*node);**

**struct tree \*root = NULL, \*node = NULL, \*temp = NULL;**

**void main()**

**{**

**int choice = 0;**

**while(choice != 5)**

**{**

**clrscr();**

**printf("\n MAIN MENU (Operations on Binary Search Tree)");**

**printf("\n 1. INSERT or CREATE");**

**printf("\n 2. PRE-ORDER TRAVERSAL (DISPLAY)");**

**printf("\n 3. IN-ORDER TRAVERSAL (DISPLAY)");**

**printf("\n 4. POST-ORDER TRAVERSAL (DISPLAY)");**

**printf("\n 5. EXIT");**

**printf("\nEnter your choice: ");**

**scanf("%d", &choice);**

**switch(choice)**

**{**

**case 1:**

**insert();**

**break;**

**case 2:**

**preorder(root);**

**break;**

**case 3:**

**inorder(root);**

**break;**

**case 4:**

**postorder(root);**

**break;**

**case 5:**

**break;**

**default:**

**printf("\n Invalid choice");**

**}**

**printf("\n Press any key to continue...");**

**getch();**

**}**

**}**

**void insert()**

**{**

**struct tree \*newnode = NULL;**

**int num = 0;**

**printf("\n Enter a number: ");**

**scanf("%d", &num);**

**newnode = (struct tree \*) malloc(sizeof(struct tree));**

**newnode -> value = num;**

**newnode -> left = NULL;**

**newnode -> right = NULL;**

**if(root == NULL)**

**{**

**root = newnode;**

**}**

**else**

**{**

**node = root;**

**while(node != NULL)**

**{**

**temp = node;**

**if(num < node -> value)**

**{**

**node = node -> left;**

**}**

**else**

**{**

**node = node -> right;**

**}**

**}**

**node = temp;**

**if(num < node -> value)**

**{**

**node -> left = newnode;**

**}**

**else**

**{**

**node -> right = newnode;**

**}**

**}**

**}**

**/\* ROOT - LEFT - RIGHT \*/**

**void preorder(struct tree \*node)**

**{**

**if(node != NULL)**

**{**

**printf("%d ", node -> value);**

**preorder(node -> left);**

**preorder(node -> right);**

**}**

**}**

**/\* LEFT - ROOT - RIGHT \*/**

**void inorder(struct tree \*node)**

**{**

**if(node != NULL)**

**{**

**inorder(node -> left);**

**printf("%d ", node -> value);**

**inorder(node -> right);**

**}**

**}**

**/\* LEFT - RIGHT - ROOT \*/**

**void postorder(struct tree \*node)**

**{**

**if(node != NULL)**

**{**

**postorder(node -> left);**

**postorder(node -> right);**

**printf("%d ", node -> value);**

**}**

**}**