21/12/20

LAB PROGRAM - 10

NAME : SAIPRAVEEN MARNI
USN : 1BM19CS138

```
#include <stdio.h>
#include <malloc.h>
  struct node {
          struct node *left;
           int value;
          struct node *right;
  };
  typedef struct node *NODE;
     NODE getnode() {
         NODE temp;
      temp = (NODE)malloc (sizeof(struct node));
         return temp;
     }
     NODE insert(NODE root) {
         int value;
       NODE temp, cur, prev;
       temp = getNode();
      printf("Enter the value :\n");
       scanf("%d\n", &value);
       temp->value = value;
       temp->left = NULL;
       temp->right = NULL;
        if(root == NULL) {
           return temp;
       }
        cur = root;
        prev = NULL;
        while(cur != NULL) {
              prev = cur;
           if(value < cur->value) {
             cur = cur->left;
```

```
    }
    else {
        curr = curr->right;
    }
}
if (value < prev->value) {
    prev->left = temp;
}
else {
    prev->right = temp;
}
return root;
}

void display (NODE root, int i) {
    int j;
    if (root != NULL) {
    display (root->right, i+1);
    for( j=0; j<i; j++) {
        printf(" ");
    }
    printf("%d \n", root->value);
    display(root->left, i+1);
}
}

void preOrder (NODE root){
    if (root == NULL) {
        return;
    }
    printf ("%d ", root->value);
    preOrder (root->left);
    preOrder (root->right);
}
```

```
void inOrder (NODE root) {
        if (root == NULL) {
        return ;
}
   inOrder (root -> left) ;
   printf ("%d", root -> value) ;
   inOrder (root -> right) ;
}

void postOrder (NODE root) {
        if (root == NULL) {
            return ;
    }
    postorder (root -> left) ;
    postOrder (root -> right) ;
    printf ("%d", root -> value) ;
}

int main() {
    int ch;
    NODE root = NULL ;
  while(1) {
printf ("Enter the choice : \t 1 - Insert \t 2 - Display \t 3 - Preorder
        \t 4 - Inorder \t 5 - Postorder \t 6 - Exit \n");
    scanf ("%d", &ch)
    switch (ch) {
  case 1 :
        root = insert (root) ;
          break ;
  case 2 :
        if (root == NULL) {
      printf ("Tree is empty \n");
    } else {
```

```
            display (root, 0);
        }
        break;
    case 3:
        if (root == NULL) {
            Printf ("Tree is Empty \n");
        }
        else {
            preOrder (root);
            printf (\n);
        }
        break;
    case 4:
        if (root == NULL) {
            printf ("Tree is Empty \n");
        }
        else {
            inOrder (root);
            printf (\n);
        }
        break;
    case 5:
        if (root == NULL) {
            printf ("Tree is Empty \n");
        }
        else {
            postOrder (root);
            printf ("\n");
        }
    case 6:
        return 0;
    }
  }
}
```