12/2/24

Q1) ① Construct binary search tree
② Traverse the tree using inorder, postorder & preorder
③ Display the elements in tree.

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

struct node {
    int data;
    struct node * left;
    struct node * right;
};

struct node * tree = NULL;
struct node * insertElement (struct node);
void preorderTraversal (struct node *);
void inorderTraversal (struct node *);
void postorderTraversal (struct node *);

```
void disp (struct node *);
void main () {

    int option, val;
    while (1) {
    printf ("\n *** Main Menu ***\n");
    printf ("\n 1. Insert Element");
    printf ("\n 2. Preorder Traversal");
    printf ("\n 3. Inorder Traversal");
    printf ("\n 4. Postorder Traversal");
    printf ("\n 5. Display");
    printf ("\n 6. Exit");
    scanf ("%d", &option);
    switch (option) {

    case 1: printf ("\n Enter value new node:");
            scanf ("%d", &val);
            tree = insert Element (tree, val);
            break;
```

```
case 2: printf("\n Elements of tree:\n");
        preorderTraversal(tree);
        break;

case 3: printf("Elements in inorder are:\n");
        inorderTraversal(tree);
        break;

case 4: printf("Elements in postorder:\n");
        postorderTraversal(tree);
        break;

case 5: printf("\n Tree Elements:\n");
        disp(tree);

case 6: exit(0)
default: printf("Invalid input");
    }
  }
}.
```

```
struct node * insertElement (struct node*
tree, int val) {

struct node * ptr, * nodeptr, * parentptr;
ptr = (struct node *) malloc (sizeof (struct node));
    ptr -> data = val;
    ptr -> left = NULL;
    ptr -> right = NULL;
    if (tree == NULL) {
        tree = ptr;
        tree -> left = NULL;
        tree -> right = NULL;
    }
    else {
        parentptr = NULL;
        nodeptr = tree;
        while (nodeptr != NULL) {
            parentptr = nodeptr;
            if (val < nodeptr -> data)
                nodeptr = nodeptr -> left;
            else
```

```
        nodeptr = nodeptr -> right;
    }
    if (val < parentptr -> data)
        parentptr -> left = ptr;
    else
        parentptr -> right = ptr;
    }
    return true
}

void preorderTraversal (struct node *tree)
    if (tree != NULL){
    printf ("%d", tree -> data);
    preorderTraversal (tree -> left);
    preorderTraversal (tree -> right);
    }
}.
```

```
void inorderTraversal (struct node * tree) {
    if (tree != NULL) {
        preorderTraversal (tree -> left);
        postorderTraversal (tree -> right);
        printf ("%d", tree -> data);
    }
}

void disp (struct node * tree) {
    if (tree != NULL) {
        disp (tree -> left);
        printf ("%d\t", tree -> data);
        disp (tree -> right);
    }
}
```

O/P :

" Main Menu "

1. Insert Element
2. Preorder Traversal
3. Inorder Traversal
4. Postorder Traversal
5. Display Elements
6. Exit

1.

Enter value of new node: 5

1 (Inserted)

Enter value of new node: 3

1 (Inserted)

Enter value of new node: 8.

1 (Inserted)

Enter value of new node = 2

5

The elements of tree are:

2 3 5 8,