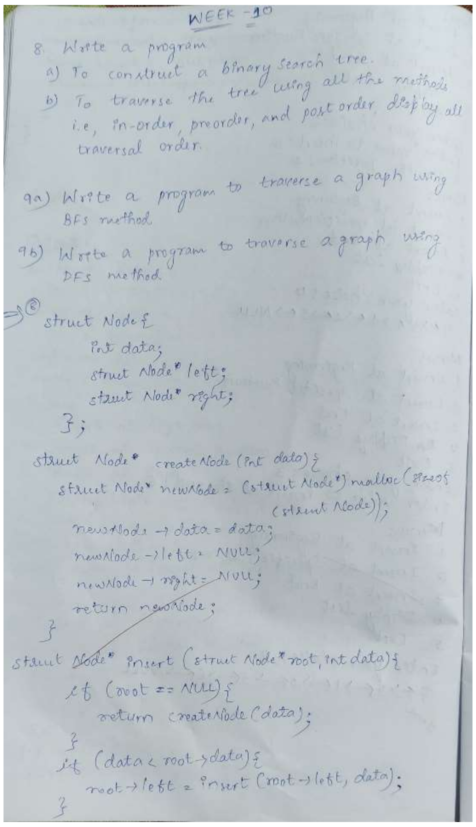
## Program 9

Write a program
8a) To construct a binary Search tree.
8b) To traverse the tree using all the methods i.e., in-order, preorder and post order, display all traversal order

Observation:



WEEK -10

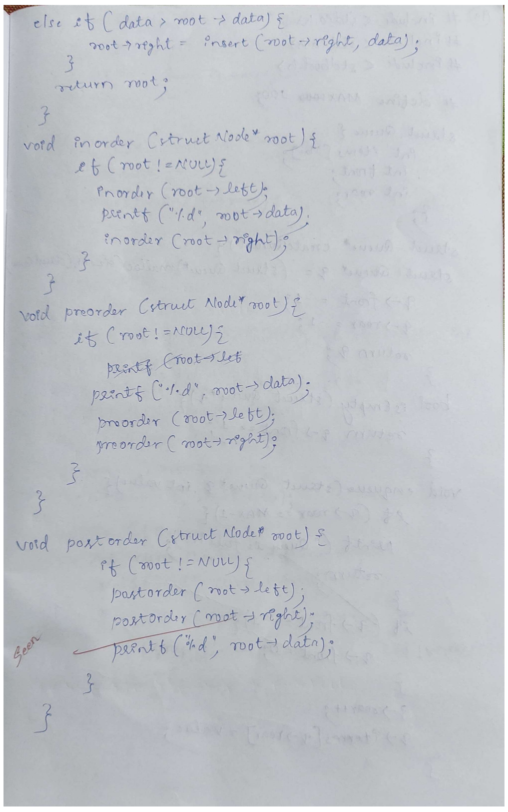
8. Write a program
a) To construct a binary Search tree.
b) To traverse the tree using all the methods i.e, in-order, preorder, and post order, display all traversal order.

9a) Write a program to traverse a graph using BFS method.

9b) Write a program to traverse a graph using DFS method.

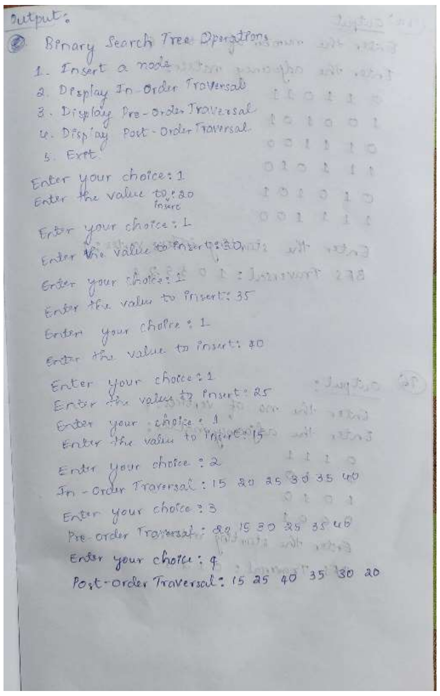
⑧

```
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);
  }
```



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

Seen



Output:

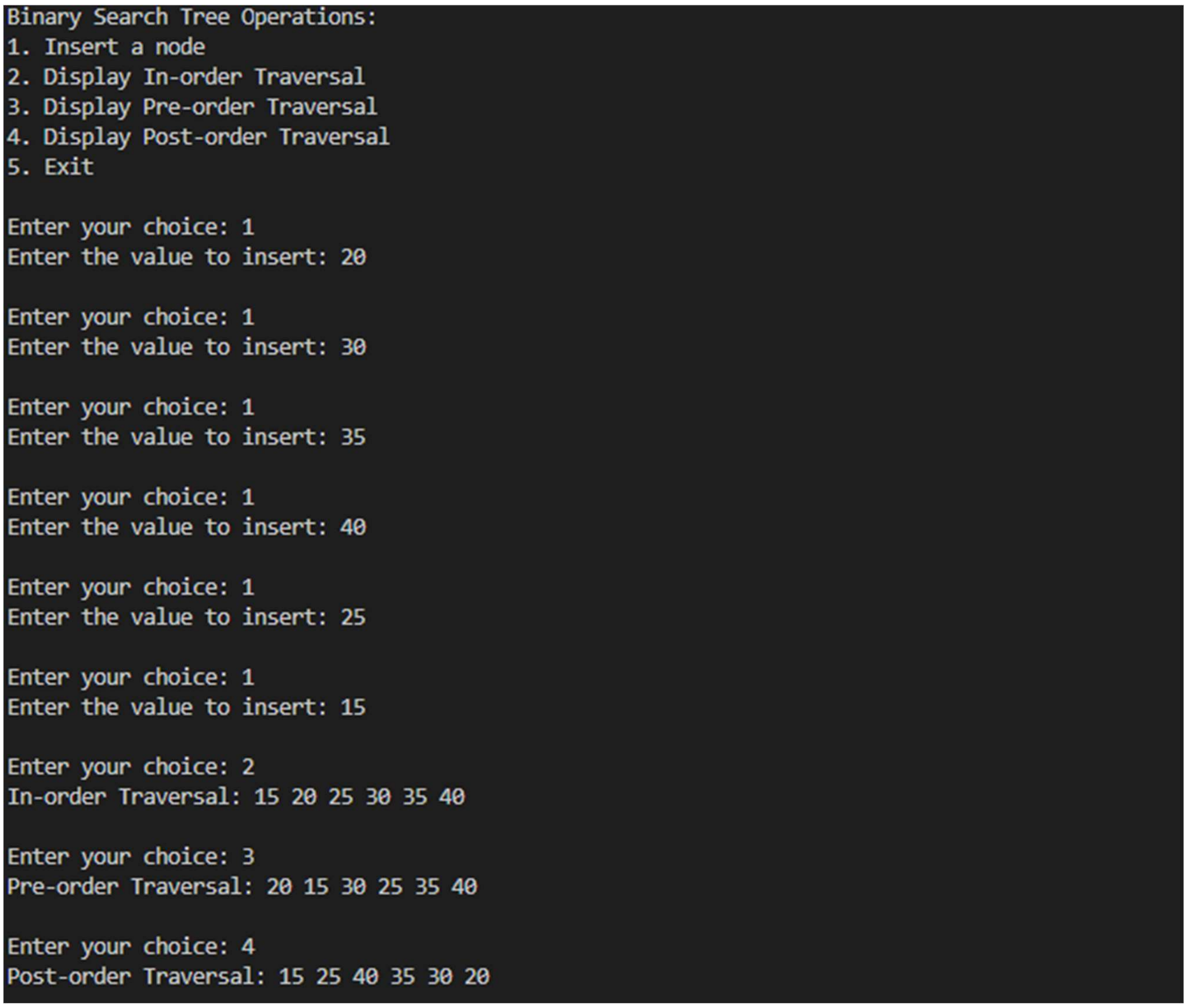
```
⑧ Binary Search Tree Operations
   1. Insert a node
   2. Display In-Order Traversal
   3. Display Pre-Order Traversal
   4. Display Post-Order Traversal
   5. Exit
Enter your choice: 1
Enter the value to insert: 20
Enter your choice: 1
Enter the value to insert: 30
Enter your choice: 1
Enter the value to insert: 35
Enter your choice: 1
Enter the value to insert: 40
Enter your choice: 1
Enter the value to insert: 25
Enter your choice: 1
Enter the value to insert: 15
Enter your choice: 2
In-Order Traversal: 15 20 25 30 35 40
Enter your choice: 3
Pre-order Traversal: 20 15 30 25 35 40
Enter your choice: 4
Post-order Traversal: 15 25 40 35 30 20
```

Code:

```c
#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);
    } else if (data > root->data) {
        root->right = insert(root->right, data);
    }
    return root;
}


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


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

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


int main() {
    struct Node* root = NULL;
    int choice, value;

    printf("Binary Search Tree Operations:\n");
    printf("1. Insert a node\n");
    printf("2. Display In-order Traversal\n");
    printf("3. Display Pre-order Traversal\n");
    printf("4. Display Post-order Traversal\n");
    printf("5. Exit\n");

    while (1) {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the value to insert: ");
                scanf("%d", &value);
                root = insert(root, value);
                break;
            case 2:
                printf("In-order Traversal: ");
                inorder(root);
                printf("\n");
                break;
            case 3:
                printf("Pre-order Traversal: ");
                preorder(root);
                printf("\n");
                break;
            case 4:
                printf("Post-order Traversal: ");
                postorder(root);
                printf("\n");
                break;
            case 5:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```

Output:



```
Binary Search Tree Operations:
1. Insert a node
2. Display In-order Traversal
3. Display Pre-order Traversal
4. Display Post-order Traversal
5. Exit

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

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

Enter your choice: 1
Enter the value to insert: 35

Enter your choice: 1
Enter the value to insert: 40

Enter your choice: 1
Enter the value to insert: 25

Enter your choice: 1
Enter the value to insert: 15

Enter your choice: 2
In-order Traversal: 15 20 25 30 35 40

Enter your choice: 3
Pre-order Traversal: 20 15 30 25 35 40

Enter your choice: 4
Post-order Traversal: 15 25 40 35 30 20
```