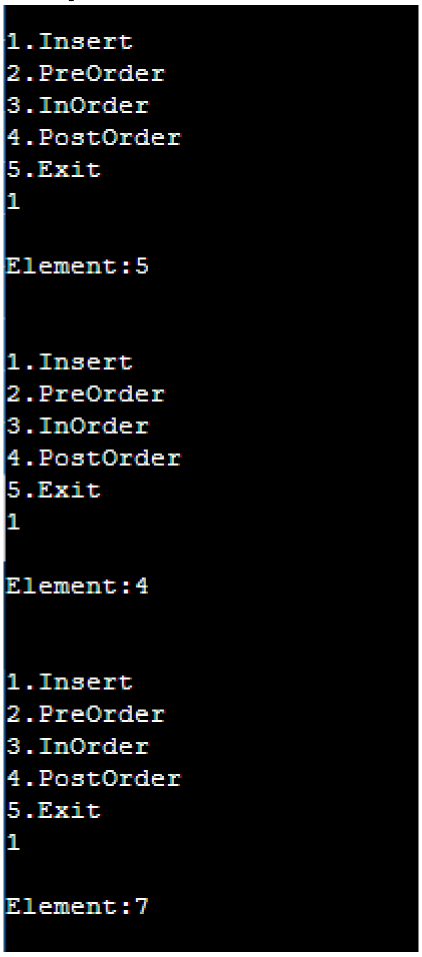
10. Write a program a) To construct a binary Search tree. b) To traverse the tree using all the methods i.e., in-order, preorder and post order c) To display the elements in the tree.

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

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

node getnode(int item)
{
   node temp = (node)malloc(sizeof(struct Node));

   temp->left = NULL;
   temp->data = item;
   temp->right = NULL;

   return temp;
}

node insert(node root, int ele)
{
   if (root == NULL)
      return getnode(ele);
   else if (ele < root->data)
      root->left = insert(root->left, ele);
   else if (ele > root->data)
      root->right = insert(root->right, ele);

   return root;
}
```

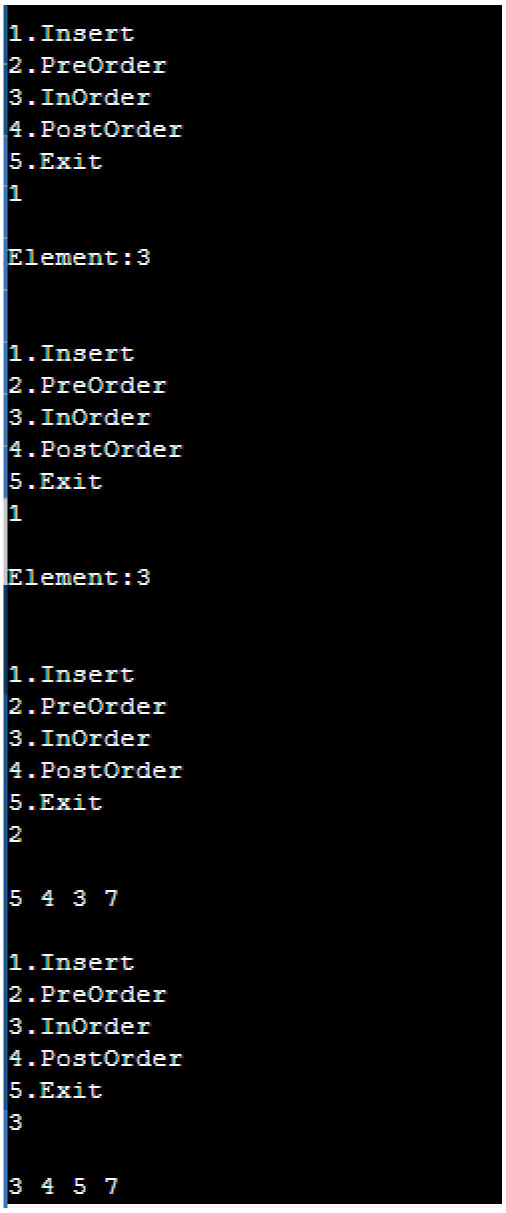
```
void inorder(node root)
{
   if (root == NULL)
      return;

   inorder(root->left);
   printf("%d ", root->data);
   inorder(root->right);
}

void preorder(node root)
{
   if (root == NULL)
      return;

   printf("%d ", root->data);
   preorder(root->left);
   preorder(root->right);
}

void postorder(node root)
{
   if (root == NULL)
      return;

   postorder(root->left);
   postorder(root->right);
   printf("%d ", root->data);
}

int main()
{
   node root = NULL;
   int e, ch = 1;
```

```
  while (ch != 5)
  {
    printf("\n\n1.Insert\n2.PreOrder\n3.InOrder\n4.PostOrder\n")
;
    printf("5.Exit\n");
    scanf("%d", &ch);
    printf("\n");

    switch (ch)
    {
    case 1:
      printf("Element:");
      scanf("%d", &e);
      root = insert(root, e);
      break;

    case 2:
      preorder(root);
      break;

    case 3:
      inorder(root);
      break;

    case 4:
      postorder(root);
      break;

    case 5:
      printf("Exiting.");
      exit(1);

    default:
      printf("Wrong input!");
```

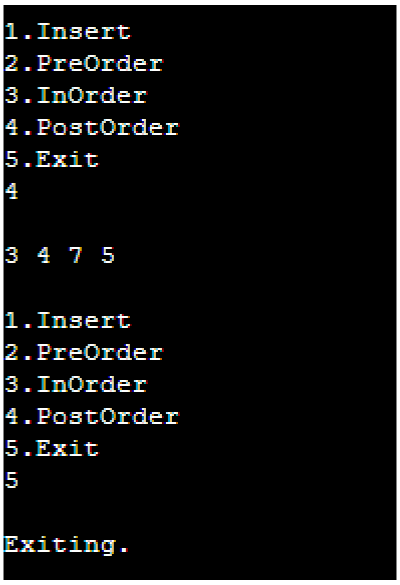
```
        }
    }
}
```
Output:



```
1.Insert
2.PreOrder
3.InOrder
4.PostOrder
5.Exit
1

Element:5


1.Insert
2.PreOrder
3.InOrder
4.PostOrder
5.Exit
1

Element:4


1.Insert
2.PreOrder
3.InOrder
4.PostOrder
5.Exit
1

Element:7
```



```
1.Insert
2.PreOrder
3.InOrder
4.PostOrder
5.Exit
1

Element:3


1.Insert
2.PreOrder
3.InOrder
4.PostOrder
5.Exit
1

Element:3


1.Insert
2.PreOrder
3.InOrder
4.PostOrder
5.Exit
2

5  4  3  7

1.Insert
2.PreOrder
3.InOrder
4.PostOrder
5.Exit
3

3  4  5  7
```



```
1.Insert
2.PreOrder
3.InOrder
4.PostOrder
5.Exit
4

3  4  7  5

1.Insert
2.PreOrder
3.InOrder
4.PostOrder
5.Exit
5

Exiting.
```