15/02/24 Week - 8 : Binary search tree

```
#include<stdio.h>
#include<stdlib.h>
struct node {
  int data;
  struct node *left;
  struct node *right;
};
struct node * NewNode(int data) {
  struct node *node = (struct node *) malloc
                      (sizeof(struct node));
  node->data = data;
  node->left = node->right = NULL;
  return node;
}
structnode *insert(struct node *root, int data)
{
  if (root == NULL)
    return NewNode(data);
  if (data <= root->data)
    root->left = insert(root->left, data);
  else
    root->right = insert(root->right, data);
  return root;
}
```

(left->root->right) 15/2/24

```
void inorder(struct node *temp) {
  if (temp == NULL)
    return;
  inorder(temp->left);
  printf("%d", temp->data);
  inorder(temp->right);
}
```

(root → left → right

```
void preorder(struct node *temp){
  if(temp==NULL)
    return;
  printf("%d", temp->data);
  preorder(temp->left);
  preorder(temp->right);
}
void postorder(struct node *temp){
  if(temp==NULL)
    return;
  postorder(temp->left);
  postorder(temp->right);
  printf("%d", temp->data);
}
int main()
{
  struct node *root=NULL;
  int data, choice;
  do{
  printf("Enter your choice");
  printf("1. Insert");
  printf("2. Inorder traversal");
  printf("3. Preorder traversal");
  printf("4. Postorder traversal");
  printf("5. Exit");
  scanf("%d", &choice);
  switch(choice)
  {
  case 1: printf("Enter the value to be inserted");
          scanf("%d", &data);
          root = insert(root, data);
          break;
  case 2: printf("Inorder traversal of binary tree");
          inorder(root);
          break;
```

(left → right → root)

```
case 3: printf("Preor
        preorder(ro
        break;
case 4: printf("Pos
        postorder(
        break;
case 5: printf("
        break;
default: printf
}
} while(choice!=5
return 0;
}
```

output

main menu

1. Insert

2. inorder trave

3. postorder tr

4. preorder tr

5. Exit

enter the choi

enter the val

Enter the cho

Enter the val

Enter the c

Enter the v

Enter the

Enter the v

```
case 3: printf("Preorder traversal of binary tree is \n");
        preorder(root);
        break;
case 4: printf("Postorder traversal of binary tree \n");
        postorder(root);
        break;
case 5: printf("Exit");
        break;
default: printf("Invalid choice. Please try again");
}
} while(choice != 5)
return 0;
}
```

output

Main menu

1. insert
2. inorder traversal
3. postorder traversal
4. preorder traversal
5. Exit

Enter the choice: 1
Enter the value to be inserted: 100

Enter the choice: 1
Enter the value to be inserted 20

Enter the choice: 1
Enter the value to be inserted 200

Enter the choice: 1
Enter the value to be inserted: 10

Enter the choice: 1
Enter the value to be inserted: 30

Enter the choice: 1
Enter the value to be inserted: 150

Enter the choice: 1
Enter the value to be inserted: 300

Enter the choice: 2
Inorder traversal: 10 20 30 100 150 200 300

Enter the choice: 3
Postorder traversal: 10 30 20 150 300 200

Enter the choice: 4
Preorder traversal: 100 20 10 30 200 150 3

Split linked list in parts

```
typedef struct listNode lnode;
int get_len (lnode *head)
{
  int n=0;
  while(head)
  {
    n++;
    head = head->next;
  }
  return n;
}
struct listNode ** splitListToParts(struct listNode
*head, int k, int *returnSize )
{
  int n = get_len(head), elems, i, j;
  *returnSize = k;
  lnode **list = (lnode **)calloc(k, sizeof(lnode*));
    *t = head;
  if(n > k)
  {
    for(i=0; i<k; i++)
    {
      elems = i<n%k ? n/k+1 : n/k;
      j=0;
      list[i] = head;
      t = head;
      while(j++ < elems)
      {
        t = head;
        head = head->next;
      }
      t->next = NULL;
    }
```

15/2/24.

```
case
{
for(i=0; i<n; i++)
{
list[i]=head;
head=head->next;
list[i]->next=NULL;
}
}
return list;
}
```

OIP

IIP: head = [1,2,3]
K = 5

Output: ([1],[2],[3],[],[])

IIP head = [1,2,3,4,5,6,7,8,9,10]
K = 3

Output = ([1,2,3,4],[5,6,7],[8,9,10])