Lab 10

Binary Search Tree

```
#include<stdio.h>
#include<stdlib.h>
# include < process.h>
struct node
{
  int info;
  struct node *rlink;
  struct node *llink;
};

typedef struct node *NODE;
NODE getNODE()
 {
  x = (NODE) malloc(sizeof(struct node));
 if (x = NULL)
 {
  printf(" mem full\n");
   exit(0);
 }
  return x;
}
void freenode(NODE x)
{
  free(x);
}
```

```
NODE insert (NODE root, int item)
{
   NODE temp, cur, prev;
    temp = getnode ();
    temp -> rlink = NULL;
    temp -> llink = NULL;
    temp -> info = item;
    if ( root == NULL)
          return temp;
    prev = NULL;
     cur = root;
  while( cur != NULL)
  {
   prev = cur;
 cur (item < cur -> info)? cur -> llink.
 }      : cur -> rlink;
 if (item < prev -> info )
      prev -> llink = temp;
 else
      prev -> rlink = temp;
 return root;
}
```

```
void display (NODE root, int i)
{
 int j;
 if (root! = NULL)
 {
   display ( root ->rlink, i+1);
   for (j=0; j<i; i++)
         printf( " ");
   printf( "%d \n", root ->info);
       ~~display (root, int item)~~.
       display (root -> llink, i+1);
 }
}

NODE delete (NODE root, int item)
{
 NODE cur, parent, q, suc;
 if (root == NULL)
{
 printf ("empty\n");
 return root;
}
 parent =NULL
 cur = root;
```

```
while( curr != NULL && item != curr->info)
{
  parent = curr;
  curr = (item < curr->info) ? curr->llink
           : curr->rlink;
}
if (curr == NULL)
{
  printf (" not found\n");
  return root;
}
if (curr->llink == NULL)
      q = curr->rlink;
else if ( curr->rlink == NULL)
      q = curr->llink;
else
{
  suc = curr -> rlink;
  while (suc -> llink != NULL)
        suc = suc -> llink;
  suc -> llink = curr -> llink;
  q = curr -> rlink;
}
```

```
if (parent == NULL)
    return q;
if ( curr == parent -> llink)
    parent -> llink = q;
else
    parent -> rlink = q;
free node( curr)
return root;
}

void preOrder( NODE root)
{
  if (root != NULL)
  {
    printf (" %d\n", root -> info);
    preorder (root -> llink);
    preorder (root -> rlink);
  }
}
void postOrder (NODE root)
{ if (root != NULL)
  postorder ( root -> llink);
  postOrder( root -> rlink);
  printf (" %d\n", root -> info);
}
```

```
void inorder(NODE root)
{ if (root != NULL)
  {
   inorder ( root -> llink);
   printf (" %d\n", root->info);
   inorder ( root -> rlink);
  }
}

void main ( )
{
 int item , choice;
 NODE root = NULL;
 for (; ;)
 {
 printf(" \n 1. insert \n 2. display \n
          3. pre \n 4. post \n 5. in \n 6. delete
          \n 7. exit \n");
 scanf (" %d", & choice);
 switch (choice)
 {
  case 1: printf ( "Enter item \n");
          scanf (" %d", & item);
          root = insert ( root, item);
          break;
```

```
case 2: display( root, 0);
        break;
case 3: preorder( root);
          break;
case 4; post order( root);
          break;
case 5; inorder( root)
           break;
case 6; printf( "Enter item\n");
           Scanf ("%d", & item);
           root= delete( root, item);
           break;
default: exit (0);
           break;
```

[illegible]