Binary Search Tree

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

struct node {
 int info;   struct node *rlink;
 struct node *llink; };

typedef struct node *NODE;

NODE ~~getch~~ getnode ( ){
  NODE x;
  x=(NODE) malloc (sizeof(struct node));
  if (x==NULL){
   printf("memory full");
   exit(0); }
  return x;
}
void freenode(NODE x){
  free(x); }

NODE insert (NODE root, int item){
 { NODE temp, cur, prev;
  temp = getnode();
  temp→ info = item;
  temp→ llink = NULL;
  temp→ rlink = NULL;
  if(root == NULL)
   return temp;
  prev = NULL;
  cur = root;
```

```
while (curr != NULL) {
~~curr~~ prev = curr;
curr = (item < curr -> info) ? curr -> llink : curr -> 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; j++)
      printf("    ");
   printf("%d," root -> temp);
  printf("    ");
   display (root -> link, i+1); }}
NODE delete (NODE root, int item) {
  NODE curr, ~~prev~~, parent , q , suc ;
  if (root == NULL) {
     printf("Empty");
      return root; }
  parent = NULL;
  curr = root;
while (curr != NULL && item != curr -> info) {
   parent = curr;
 curr = (item < curr -> info) ? curr -> llink : curr -> rlink;
 }
```

```
~~if (curr → l~~
if (curr = = NULL)
{ printf("Element not found");
  return root; }
if ( curr→llink = = NULL)
  q = curr→rlink;
void preorder (NODE root)
 { if ( root != NULL){
  printf("%d", root→info);
 preorder(root→llink);
 preorder ( root →rlink); }}
void postorder (NODE root){
  { if(root != NULL){
    postorder (root → llink);
    postorder ( root →rlink);
   printf ("%d", root → info); }}
void inorder(NODE root){
  if (root != NULL){
   inorder ( root → llink);
   printf ("%d", root → info);
   inorder ( root→ rlink); }}
void main(){
 int ~~info~~, item, choice;
 NODE root = NULL;
 for(;;){
 printf("Enter your choice. \n1. insert \n2 display
 \n3 preorder \n4 postorder \n5 inorder
   \n6 delete");
 scanf("% d",&choice);
  switch(ch){
```

```
case 1 : printf("Enter the item");
         scanf(" %d", &item);
~~case 2 :~~ root = insert(root, item);
         break;
case 2 : display(root, 0);
         break;
case 3 : preorder(root); break;
case 4 : postorder( root); break;
case 5 : inorder (root); break;
case 6 : printf ("Enter item to be deleted");
       scanf("%d", &item);
       root = delete(root, item);
       break;
  }}}
```