BINARY SEARCH TREE

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct node
  { int info;
    struct node *rlink;
    struct node *llink;
  };
typedef struct node *NODE;
  NODE getnode()
  {
    NODE x;
    x = (NODE) malloc (sizeof (strucenode));
    if (x==NULL)
      { pf ("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 (cue != NULL)
  { prev = cue;
  cue = (item < cue->info) ? cue->llink : cue->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++)
        pf ("  ");
  pf ("%d\n", root->info);
display (root->llink, i+1);
  }
}
void preorder (NODE root)
  { if (root != NULL)
    {
  pf ("%d\t", root->info);
  preorder(root->llink);
  preorder(root->rlink);
    }
}
```

```
void postorder (NODE root)
  {
    if (root != NULL)
    {
      postorder (root -> llink);
      postorder (root -> rlink);
      printf ("%d\t", root->info);
    }
  }
void inorder (NODE root)
  {
    if (root != NULL)
    {
      inorder (root -> llink);
      pf ("%d\t", root->info);
      inorder (root->rlink);
    }
  }
void main()
  { int item, choice;
    NODE root = NULL;
    for (;;)
    {
pf ("1. insert 2. disp 3. pre 4. post 5. in 6. exit");
  pf ("Enter your choice");
    sf ("%d", &choice);
      switch (choice)
    { case 1: pf ("Enter the item\n");
              sf ("%d", &item);
              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;
default : exit(0);
    break;
  }
 }
}
```