DATE
PAGE

WAP

a) To construct a binary search tree

b) To traverse the tree using inorder, preorder & postorder

c) display elements of tree.

```
#include <stdio.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 (struct node));
          if (x == NULL)
          {
          printf("mem full\n");
             exit(0);
           }
            return x;
          }
```

Scanned with CamScanner

```
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;
   }
```

Scanned with CamScanner

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

void postorder (Node root)
{
if (root != NULL)
{
postorder (root -> llink);
postorder (root->rlink);
printf (" %od\n", root -> info);
}
}
```

Scanned with CamScanner

```
void inorder (Node root)
{
if (root != NULL)
  {
  inorder (root->llink);
  printf (" %d\n", root->info);
  inorder( root->rlink);
  }
}
int main()
{
int item, choice;
Node root = NULL;
 for (;;) {
printf (" \n1. insert \n 2. display \n 3. preorder
\n 4. postorder \n 5. inorder \n 6. exit \n);

printf ( " Enter choice \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;
```

Scanned with CamScanner

```
case 3: preorder(root);
        break;

case 4: postorder(root);
        break;

case 5: inorder(root);
        break;

default : exit(0);
          break;
    }
  }
}
```

Scanned with CamScanner

Expected output :-

1. insert
2. display
3. preorder
4. postorder
5. inorder
6. exit

Enter choice :
1
Enter item :
15

Enter choice :
1
Enter item
2

Enter choice
1
Enter item
67

Scanned with CamScanner

```
Enter   choice :
1
Enter item
67

Enter   choice :
1
Enter  item
90

Enter   choice
1
Enter  item
34

Enter   choice
2
        90
      67
        34
    15
      2

Enter   choice :
5
2
15
34
67
90
```

Scanned with CamScanner

Enter choice :
3

15

2
67
34
90

Enter choice
4

2
34
90
67
15

Scanned with CamScanner