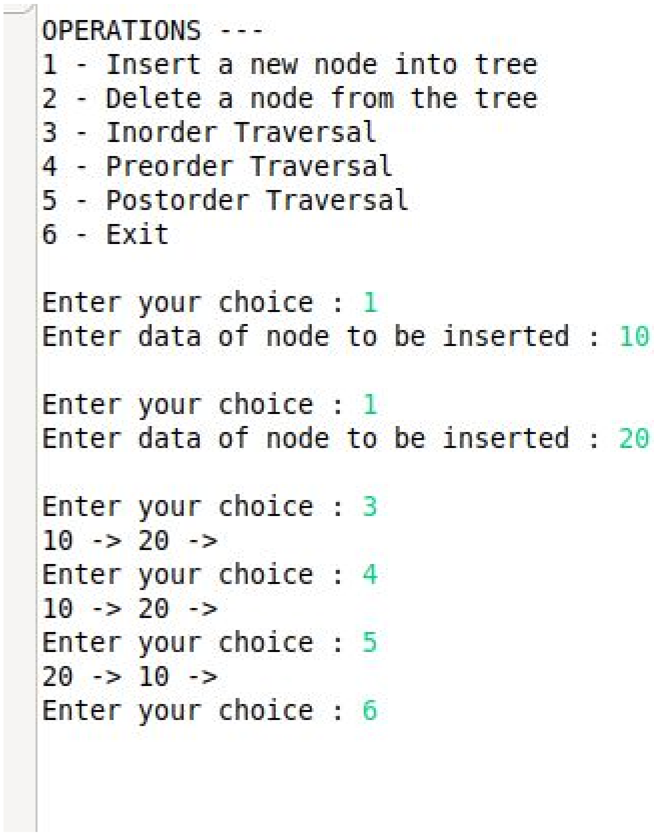
1. Create a binary search tree with the following operations
  1.1) Insert a new node
  1.2.) Inorder traversal.
  1.3.) Preorder traversal.
  1.4.) Postorder traversal.
  1.5.) Delete a node.

Ans :

```
      #include <stdio.h>
#include <stdlib.h>
struct btnode
{
    int value;
    struct btnode *l;
    struct btnode *r;
}*root = NULL, *temp = NULL, *t2, *t1;
void delete1();
void insert();
void delete();
void inorder(struct btnode *t);
void create();
void search(struct btnode *t);
void preorder(struct btnode *t);
void postorder(struct btnode *t);
void search1(struct btnode *t,int data);
int smallest(struct btnode *t);
int largest(struct btnode *t);
int flag = 1;
void main()
{
    int ch;

    printf("\nOPERATIONS ---");
    printf("\n1 - Insert a new node into tree\n");
    printf("2 - Delete a node from the tree\n");
    printf("3 - Inorder Traversal\n");
    printf("4 - Preorder Traversal\n");
    printf("5 - Postorder Traversal\n");
    printf("6 - Exit\n");
    while(1)
    {
        printf("\nEnter your choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            insert();
            break;
        case 2:
            delete();
            break;
        case 3:
            inorder(root);
            break;
        case 4:
            preorder(root);
            break;
        case 5:
            postorder(root);
            break;
        case 6:
            exit(0);
        default :
            printf("Wrong choice, Please enter correct choice  ");
            break;
        }
    }
}
void insert()
{
    create();
    if (root == NULL)
        root = temp;
    else
        search(root);
}
void create()
{
    int data;

    printf("Enter data of node to be inserted : ");
    scanf("%d", &data);
    temp = (struct btnode *)malloc(1*sizeof(struct btnode));
    temp->value = data;
    temp->l = temp->r = NULL;
}
void search(struct btnode *t)
{
    if ((temp->value > t->value) && (t->r != NULL))
        search(t->r);
    else if ((temp->value > t->value) && (t->r == NULL))
        t->r = temp;
    else if ((temp->value < t->value) && (t->l != NULL))
        search(t->l);
    else if ((temp->value < t->value) && (t->l == NULL))
        t->l = temp;
}
void inorder(struct btnode *t)
{
    if (root == NULL)
    {
        printf("No elements in a tree to display");
        return;
    }
    if (t->l != NULL)
        inorder(t->l);
    printf("%d -> ", t->value);
    if (t->r != NULL)
        inorder(t->r);
}
void delete()
{
    int data;

    if (root == NULL)
    {
        printf("No elements in a tree to delete");
        return;
    }
    printf("Enter the data to be deleted : ");
    scanf("%d", &data);
    t1 = root;
    t2 = root;
    search1(root, data);
}
void preorder(struct btnode *t)
{
    if (root == NULL)
    {
        printf("No elements in a tree to display");
        return;
    }
    printf("%d -> ", t->value);
    if (t->l != NULL)
        preorder(t->l);
    if (t->r != NULL)
        preorder(t->r);
}
void postorder(struct btnode *t)
{
    if (root == NULL)
    {
        printf("No elements in a tree to display ");
        return;
    }
    if (t->l != NULL)
        postorder(t->l);
    if (t->r != NULL)
        postorder(t->r);
    printf("%d -> ", t->value);
}
void search1(struct btnode *t, int data)
{
    if ((data>t->value))
    {
        t1 = t;
        search1(t->r, data);
    }
    else if ((data < t->value))
    {
        t1 = t;
        search1(t->l, data);
    }
    else if ((data==t->value))
    {
        delete1(t);
    }
}
void delete1(struct btnode *t)
{
    int k;
       if ((t->l == NULL) && (t->r == NULL))
    {
        if (t1->l == t)
        {
            t1->l = NULL;
        }
        else
        {
            t1->r = NULL;
        }
        t = NULL;
        free(t);
        return;
    }
   else if ((t->r == NULL))
    {
        if (t1 == t)
        {
            root = t->l;
            t1 = root;
        }
        else if (t1->l == t)
        {
            t1->l = t->l;

        }
        else
        {
            t1->r = t->l;
        }
        t = NULL;
        free(t);
        return;
    }
    else if (t->l == NULL)
    {
        if (t1 == t)
        {
            root = t->r;
            t1 = root;
        }
        else if (t1->r == t)
            t1->r = t->r;
        else
            t1->l = t->r;
        t == NULL;
        free(t);
        return;
    }
```

```
     else if ((t->l != NULL) && (t->r != NULL))
    {
        t2 = root;
        if (t->r != NULL)
        {
            k = smallest(t->r);
            flag = 1;
        }
        else
        {
            k =largest(t->l);
            flag = 2;
        }
        search1(root, k);
        t->value = k;
    }

}
int smallest(struct btnode *t)
{
    t2 = t;
    if (t->l != NULL)
    {
        t2 = t;
        return(smallest(t->l));
    }
    else
        return (t->value);
}
int largest(struct btnode *t)
{
    if (t->r != NULL)
    {
        t2 = t;
        return(largest(t->r));
    }
    else
        return(t->value);
}
```

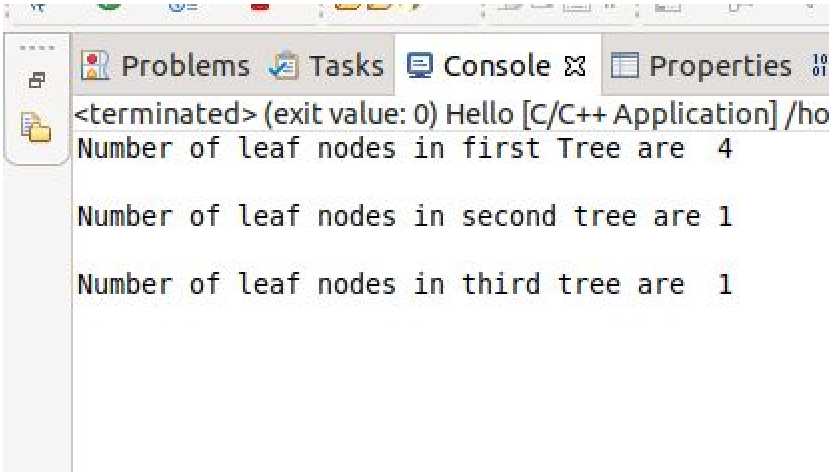


```
OPERATIONS ---
1 - Insert a new node into tree
2 - Delete a node from the tree
3 - Inorder Traversal
4 - Preorder Traversal
5 - Postorder Traversal
6 - Exit

Enter your choice : 1
Enter data of node to be inserted : 10

Enter your choice : 1
Enter data of node to be inserted : 20

Enter your choice : 3
10 -> 20 ->
Enter your choice : 4
10 -> 20 ->
Enter your choice : 5
20 -> 10 ->
Enter your choice : 6
```

2) Write a program to create a binary search tree and find the number of leaf nodes ?

Ans :

```
 #include <stdio.h>
#include <stdlib.h>
struct node
{
    int info;
    struct node* left, *right;

};
struct node* createnode(int key)
{
    struct node* newnode = (struct node*)malloc(sizeof(struct node));
    newnode->info = key;
    newnode->left = NULL;
    newnode->right = NULL;

    return(newnode);
}
int count = 0;
int leafnodes(struct node* newnode)
{

    if(newnode != NULL)
    {
        leafnodes(newnode->left);
        if((newnode->left == NULL) && (newnode->right == NULL))
        {
            count++;
        }
        leafnodes(newnode->right);
    }
    return count;

}

int main()
{
    struct node *newnode = createnode(25);
    newnode->left = createnode(27);
    newnode->right = createnode(19);
    newnode->left->left = createnode(17);
    newnode->left->right = createnode(91);
    newnode->right->left = createnode(13);
    newnode->right->right = createnode(55);

    printf("Number of leaf nodes in first Tree are\t%d\n",leafnodes(newnode));
    count = 0;

    struct node *node = createnode(1);
    node->right = createnode(2);
    node->right->right = createnode(3);
    node->right->right->right = createnode(4);
    node->right->right->right->right = createnode(5);


    printf("\nNumber of leaf nodes in second tree are\t%d\n",leafnodes(node));
    count = 0;
    struct node *root = createnode(15);
    printf("\nNumber of leaf nodes in third tree are\t%d",leafnodes(root));

    return 0;
}
```

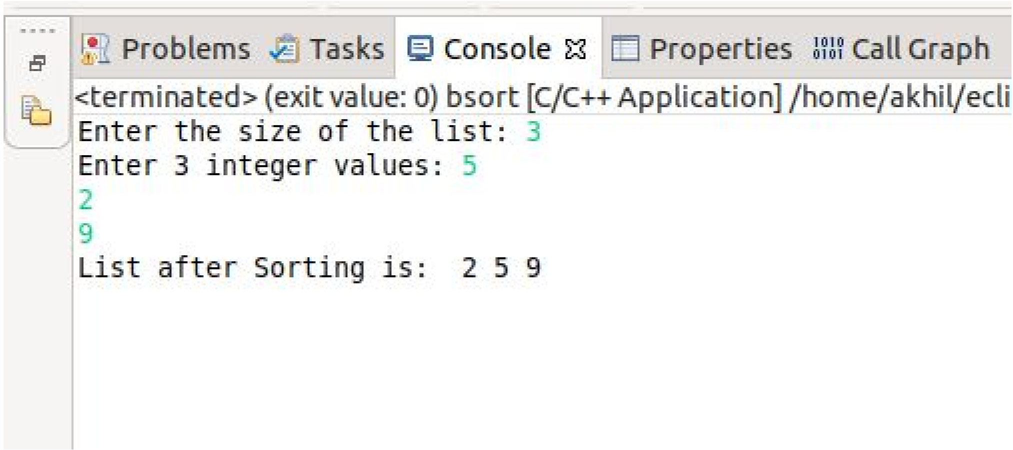


```
<terminated> (exit value: 0) Hello [C/C++ Application] /ho
Number of leaf nodes in first Tree are  4

Number of leaf nodes in second tree are 1

Number of leaf nodes in third tree are  1
```

3) Write a program to sort a set of numbers using a binary tree. ?

Ans :

```
#include<stdio.h>
#include<curses.h>
void main()
{
  int size, i, j, temp, list[100];

  printf("Enter the size of the list: ");
  scanf("%d", &size);

  printf("Enter %d integer values: ", size);
  for (i = 0; i < size; i++)
    scanf("%d", &list[i]);
  for (i = 1; i < size; i++) {
    temp = list[i];
    j = i - 1;
    while ((temp < list[j]) && (j >= 0)) {
      list[j + 1] = list[j];
      j = j - 1;
    }
    list[j + 1] = temp;
  }
  printf("List after Sorting is: ");
  for (i = 0; i < size; i++)
    printf(" %d", list[i]);


}
```

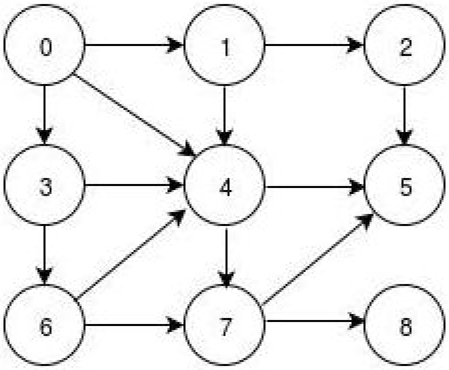


```
<terminated> (exit value: 0) bsort [C/C++ Application] /home/akhil/ecli
Enter the size of the list: 3
Enter 3 integer values: 5
2
9
List after Sorting is:  2 5 9
```

4) Write a C program to Represent any given graph and and breadth first search ?

Ans :

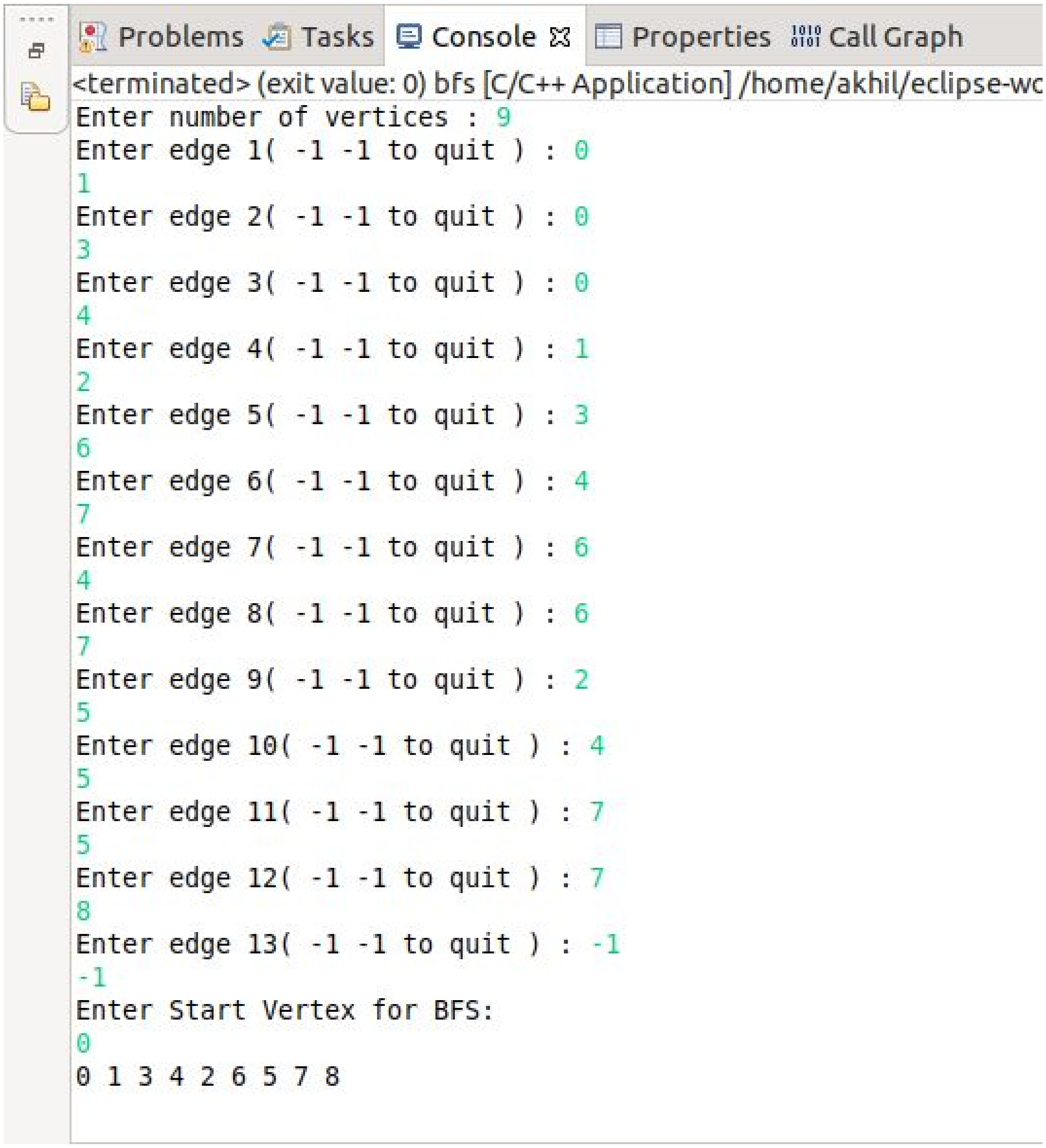
Consider below Graph as an example.



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

#define MAX 100

#define initial 1
#define waiting 2
#define visited 3

int n;
int adj[MAX][MAX];
int state[MAX];
void create_graph();
void BF_Traversal();
void BFS(int v);

int queue[MAX], front = -1,rear = -1;
void insert_queue(int vertex);
int delete_queue();
int isEmpty_queue();

int main()
{
        create_graph();
        BF_Traversal();
        return 0;
}

void BF_Traversal()
{
        int v;

        for(v=0; v<n; v++)
                state[v] = initial;

        printf("Enter Start Vertex for BFS: \n");
        scanf("%d", &v);
        BFS(v);
}

void BFS(int v)
{
        int i;

        insert_queue(v);
        state[v] = waiting;

        while(!isEmpty_queue())
        {
                v = delete_queue( );
                printf("%d ",v);
                state[v] = visited;

                for(i=0; i<n; i++)
                {
                        if(adj[v][i] == 1 && state[i] == initial)
                        {
```

```
                insert_queue(i);
                state[i] = waiting;
            }
        }
    }
    printf("\n");
}

void insert_queue(int vertex)
{
    if(rear == MAX-1)
        printf("Queue Overflow\n");
    else
    {
        if(front == -1)
            front = 0;
        rear = rear+1;
        queue[rear] = vertex ;
    }
}

int isEmpty_queue()
{
    if(front == -1 || front > rear)
        return 1;
    else
        return 0;
}

int delete_queue()
{
    int delete_item;
    if(front == -1 || front > rear)
    {
        printf("Queue Underflow\n");
        exit(1);
    }

    delete_item = queue[front];
    front = front+1;
    return delete_item;
}

void create_graph()
```

```
{
    int count,max_edge,origin,destin;

    printf("Enter number of vertices : ");
    scanf("%d",&n);
    max_edge = n*(n-1);

    for(count=1; count<=max_edge; count++)
    {
        printf("Enter edge %d( -1 -1 to quit ) : ",count);
        scanf("%d %d",&origin,&destin);

        if((origin == -1) && (destin == -1))
            break;

        if(origin>=n || destin>=n || origin<0 || destin<0)
        {
            printf("Invalid edge!\n");
            count--;
        }
        else
        {
            adj[origin][destin] = 1;
        }
    }
}
```



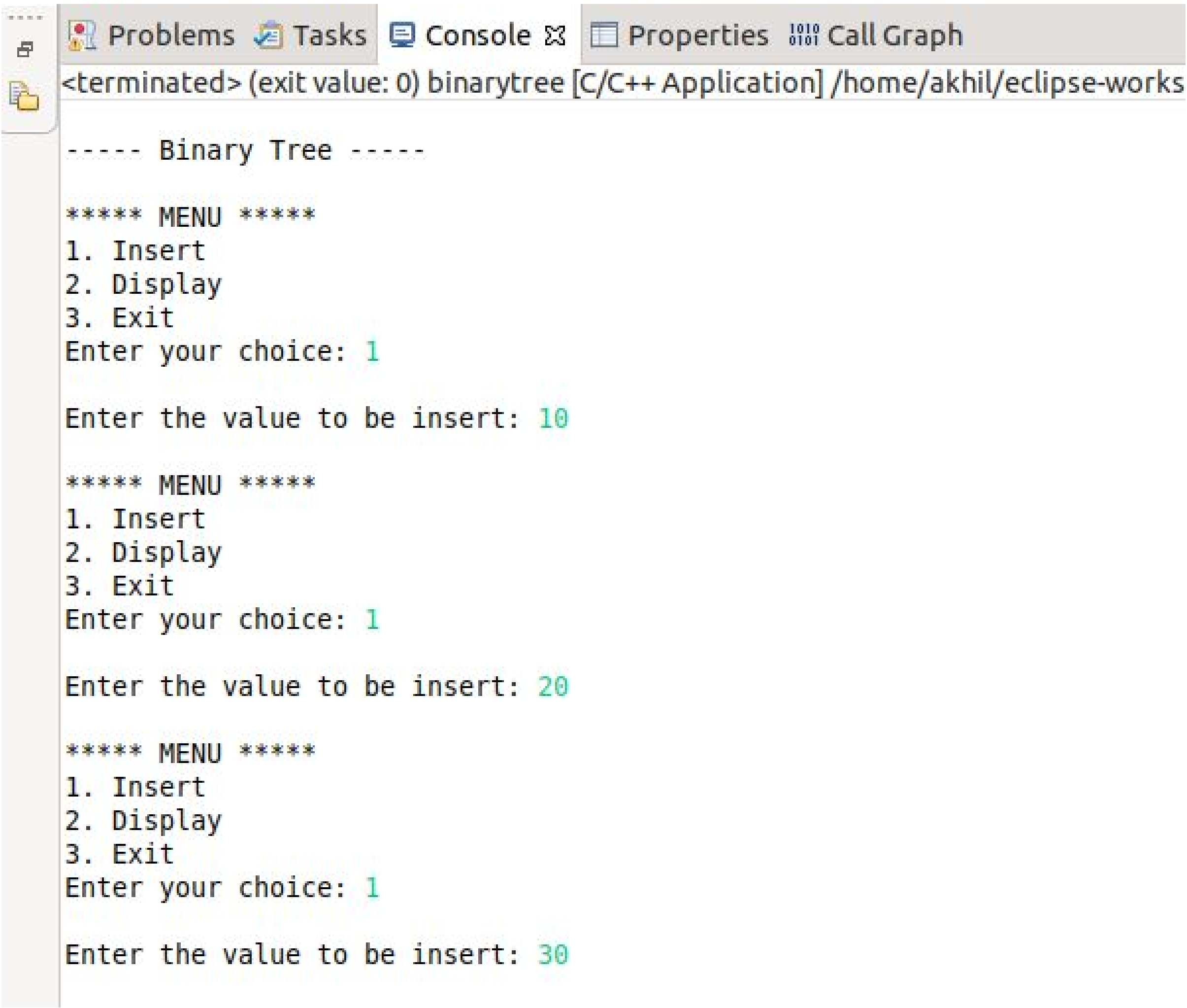
```
<terminated> (exit value: 0) bfs [C/C++ Application] /home/akhil/eclipse-wc
Enter number of vertices : 9
Enter edge 1( -1 -1 to quit ) : 0
1
Enter edge 2( -1 -1 to quit ) : 0
3
Enter edge 3( -1 -1 to quit ) : 0
4
Enter edge 4( -1 -1 to quit ) : 1
2
Enter edge 5( -1 -1 to quit ) : 3
6
Enter edge 6( -1 -1 to quit ) : 4
7
Enter edge 7( -1 -1 to quit ) : 6
4
Enter edge 8( -1 -1 to quit ) : 6
7
Enter edge 9( -1 -1 to quit ) : 2
5
Enter edge 10( -1 -1 to quit ) : 4
5
Enter edge 11( -1 -1 to quit ) : 7
5
Enter edge 12( -1 -1 to quit ) : 7
8
Enter edge 13( -1 -1 to quit ) : -1
-1
Enter Start Vertex for BFS:
0
0 1 3 4 2 6 5 7 8
```

5) Write a C Program to create a Binary Tree ?

Ans :

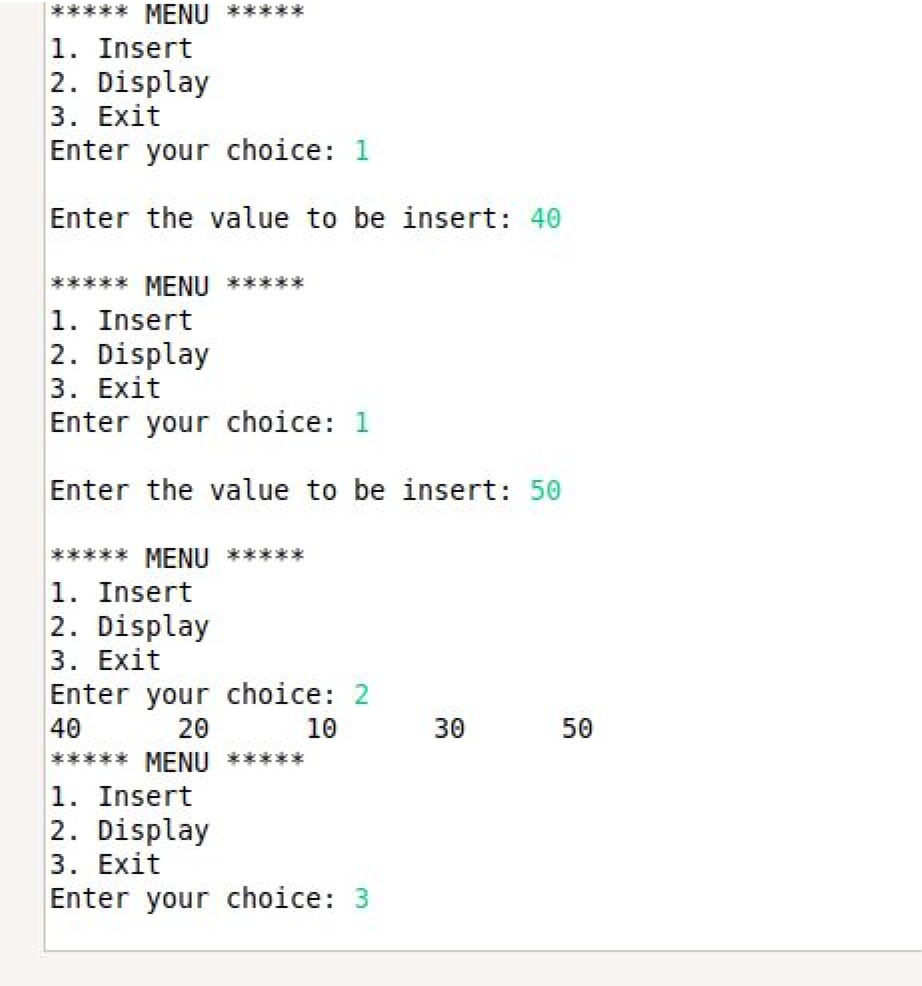
```
#include<stdio.h>
#include<curses.h>
struct Node{
   int data;
   struct Node *left;
   struct Node *right;
};

struct Node *root = NULL;
```

```
int count = 0;

struct Node* insert(struct Node*, int);
void display(struct Node*);

void main(){
   int choice, value;

   printf("\n----- Binary Tree -----\n");
   while(1){
      printf("\n***** MENU *****\n");
      printf("1. Insert\n2. Display\n3. Exit");
      printf("\nEnter your choice: ");
      scanf("%d",&choice);
      switch(choice){
      case 1: printf("\nEnter the value to be insert: ");
            scanf("%d", &value);
            root = insert(root,value);
            break;
      case 2: display(root); break;
      case 3: exit(0);
      default: printf("\nPlease select correct operations!!!\n");
      }
   }
}

struct Node* insert(struct Node *root,int value){
   struct Node *newNode;
   newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   if(root == NULL){
      newNode->left = newNode->right = NULL;
      root = newNode;
      count++;
   }
   else{
      if(count%2 != 0)
      root->left = insert(root->left,value);
      else
      root->right = insert(root->right,value);
   }
   return root;
}
void display(struct Node *root)
{
   if(root != NULL){
      display(root->left);
      printf("%d\t",root->data);
      display(root->right);
   }
}
```



<terminated> (exit value: 0) binarytree [C/C++ Application] /home/akhil/eclipse-works

```
----- Binary Tree -----

***** MENU *****
1. Insert
2. Display
3. Exit
Enter your choice: 1

Enter the value to be insert: 10

***** MENU *****
1. Insert
2. Display
3. Exit
Enter your choice: 1

Enter the value to be insert: 20

***** MENU *****
1. Insert
2. Display
3. Exit
Enter your choice: 1

Enter the value to be insert: 30
```



```
***** MENU *****
1. Insert
2. Display
3. Exit
Enter your choice: 1

Enter the value to be insert: 40

***** MENU *****
1. Insert
2. Display
3. Exit
Enter your choice: 1

Enter the value to be insert: 50

***** MENU *****
1. Insert
2. Display
3. Exit
Enter your choice: 2
40      20      10      30      50
***** MENU *****
1. Insert
2. Display
3. Exit
Enter your choice: 3
```