Example 8.1 Write a program to implement a binary tree using linked list.

Program 8.1 *Implementation of a binary tree*

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

struct bin_tree
{
 int INFO;
 struct node *LEFT, *RIGHT;
};

typedef struct bin_tree node;

node *insert(node *,int); /*Function prototype for inserting a new node*/
void display (node *); /*Function prototype for displaying the tree nodes*/
int count = 1; /*Counter for ascertaining left or right position for the
new node*/


void main()
{
 struct node *root = NULL;
 int element, choice;

 clrscr();

 /*Displaying a menu of choices*/
 while(1)
 {
 clrscr();
 printf("Select an option\n");
```

Here, the node of the tree is realised with the help of a structure declaration. The INFO field stores the node value while the LEFT and RIGHT pointers point at the left and right subtrees respectively.

Since the root node has no parents, its location is tracked with the help of a special pointer called root.

```
   printf("\n1 - Insert");
   printf("\n2 - Display");
   printf("\n3 - Exit");

   printf("\n\nEnter your choice: ");
   scanf("%d", &choice);

   switch(choice)
   {
   case 1:
   {
```



```
   printf("\n\nEnter the node value: ");
   scanf("%d",&element);
   root = insert(root,element); /*Calling the insert function for inserting
a new element into the tree*/
   getch();
   break;
   }
```

```
   case 2:
   {
   display(root); /*Calling the display function for printing the node
values*/
   getch();
   break;
   }

   case 3:
   {
   exit(1);
   break;
   }

   default:
   {
   printf("\nIncorrect choice. Please try again.");
   getch();
   break;
   }
   }
   }
  }

  node *insert(node *r, int n)
  {
   if(r==NULL)
   {
   r=(node*) malloc (sizeof(node));
   r->LEFT = r->RIGHT = NULL;
   r->INFO = n;
   count=count+1;
   }
```

The use of dynamic memory allocation ensures that memory space for a new node is allocated only at the time of its creation.

```
  r->INFO = n;
  count=count+1;
  }
  else
  {
  if(count%2==0)
  r->LEFT = insert(r->LEFT, n);
  else
  r->RIGHT = insert(r->RIGHT, n);
  }
  return(r);
```

```
}

void display(node * r)
{
 if(r->LEFT!=NULL)
 display(r->LEFT);
 printf("%d\n",r->INFO);
 if(r->RIGHT!=NULL)
 display(r->RIGHT);
}
```

Output

```
Select an option

1 - Insert
2 - Display
3 - Exit

Enter your choice: 1

Enter the node value: 1

Enter your choice: 1

Enter the node value: 2

Enter your choice: 1

Enter the node value: 3

Enter your choice: 1

Enter the node value: 4
```

```
Enter the node value: 4

Enter your choice: 1

Enter the node value: 5

Enter your choice: 1

Enter the node value: 6

Select an option

1 - Insert
2 - Display
3 - Exit

Enter your choice: 2
```



```
6
4
2
1
3
5
```

```
Select an option

1 - Insert
2 - Display
3 - Exit

Enter your choice: 3
```

Program analysis

Key Statement	Purpose
node *insert(node *,int); void display (node *);	Declares function prototypes for inserting and displaying binary tree nodes
root = insert(root,element);	Calls the *insert()* function for inserting a new node into the binary tree
display(root);	Calls the *display()* function for displaying the binary tree nodes

if(count%2==0)
r->LEFT = insert(r->LEFT, n);
else
r->RIGHT = insert(r->RIGHT, n);
	Checks the value of the *count* variable to insert the new node either in the left or right subtree