Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Technology

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

Data Structures Lab

Assignment No: 5

Name : Gourav Balaji Suram

Roll No : 39

PRN No : 12220032

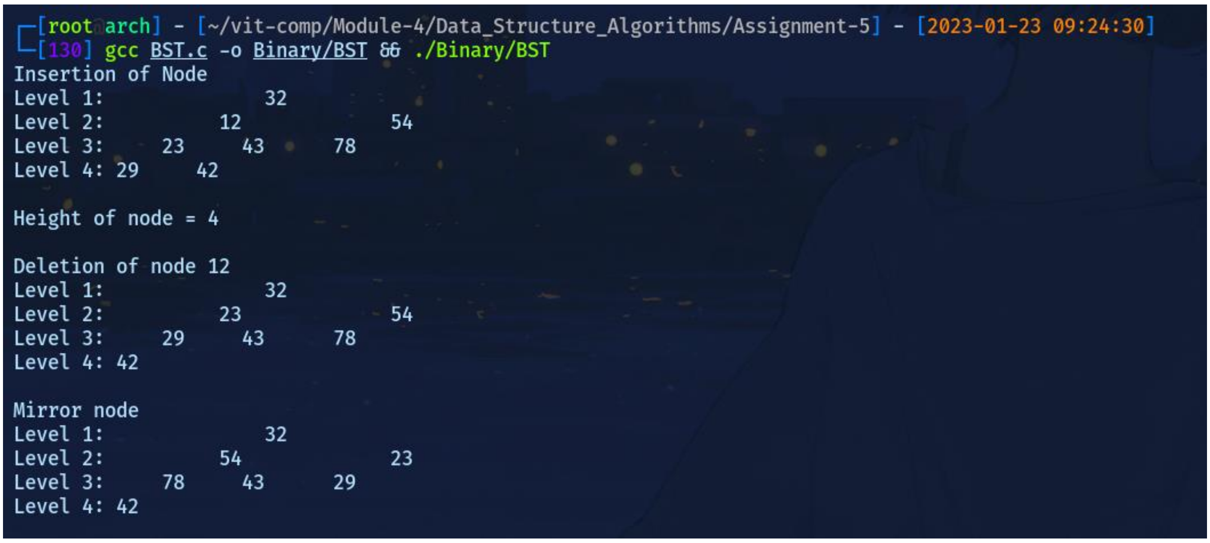
PROBLEM STATEMENT:

Create BST and Perform following Operations.

A. Insert.
B. Delete.
C. Level wise Display.
D. Mirror Image
E. Height of The Tree

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

struct node{
  int data;
  struct node* left;
  struct node* right;
};

struct node* newnode(int data){
  struct node* new = malloc(sizeof(struct node));
  new->data = data;
  new->left = new->right = NULL;
  return new;
}

struct node* insertion(struct node* head, int n){
  if(head == NULL)
    return newnode(n);
  if(n < head->data){
    head->left = insertion(head->left, n);
  }
  else if(n > head->data){
    head->right = insertion(head->right, n);
  }
  return head;
}

void printlevel(struct node* head, int n, int space){
  if(head == NULL){
    return;
  }
  if(n == 1){
    for(int i=0;i<space;i++){
      printf(" ");
    }
    printf("%d\t",head->data);
    return;
  }
  printlevel(head->left, n-1, space-n);
  printlevel(head->right, n-1, space-n);
}

int height(struct node* head){
  if(head == NULL)
    return 0;
  else{
    int lheight = height(head->left);
    int rheight = height(head->right);
    if(lheight > rheight)
      return (lheight+1);
    else
      return (rheight+1);
  }
}

void display(struct node* head){
  if(head == NULL)
    return;
  else{
    for(int i=1;i<=height(head);i++){
      int a=15-i*2;
      printf("Level %d: ",i);
      printlevel(head, i, a);
      printf("\n");
    }
  }
  printf("\n");
}

struct node* getmin(struct node* root){
  while(root->left != NULL){
    root = root->left;
  }
  return root;
}

struct node* deletion(struct node* root, int n){
  if(root != NULL){
    if(n < root->data){
      root->left = deletion(root->left, n);
    }
    else if(n > root->data){
      root->right = deletion(root->right, n);
    }
    else{
      if(root->left == NULL && root->right == NULL){
        free(root);
        return NULL;
      }
      else if(root->left != NULL && root->right == NULL){
        struct node* temp = root->left;
        free(root);
        return temp;
      }
      else if(root->right != NULL && root->left == NULL){
        struct node* temp = root->right;
        free(root);
        return temp;
      }
      struct node* temp = getmin(root->right);
      int val = temp->data;
      deletion(root, temp->data);
      root->data = val;
    }
  }
  return root;
}

void mirror(struct node* head){
  struct node* temp;
  if(head == NULL){
    return;
  }
  else{
    mirror(head->left);
    mirror(head->right);
    temp = head->left;
    head->left = head->right;
    head->right = temp;
  }
}

int main(){
  int arr[] = {32,54,12,23,78,29,43,42};
  int size = sizeof(arr)/sizeof(arr[0]);
  struct node* root = NULL;
  for(int i=0; i<size; i++){
    root = insertion(root, arr[i]);
  }
  //inorder(root);
  printf("Insertion of Node \n");
  display(root);
  int n=height(root);
  printf("Height of node = %d\n\n",n);
  printf("Deletion of node 12\n");
  deletion(root, 12);
  display(root);
  printf("Mirror node\n");
  mirror(root);
  display(root);
  return 0;
}
```

OUTPUT



```
┌─[root@arch] - [~/vit-comp/Module-4/Data_Structure_Algorithms/Assignment-5] - [2023-01-23 09:24:30]
└─[130] gcc BST.c -o Binary/BST && ./Binary/BST
Insertion of Node
Level 1:              32
Level 2:          12           54
Level 3:     23     43      78
Level 4: 29     42

Height of node = 4

Deletion of node 12
Level 1:              32
Level 2:          23           54
Level 3:     29     43      78
Level 4: 42

Mirror node
Level 1:              32
Level 2:          54           23
Level 3:     78     43      29
Level 4: 42
```