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

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

node *newnode(int data)
{
    node *ptr = (node *)malloc(sizeof(node));
    ptr->data = data;
    ptr->left = NULL;
    ptr->right = NULL;
    return (ptr);
}
void mirror(node *ptr)
{
    if (ptr != NULL)
    {
        node *temp;
        mirror(ptr->left);
        mirror(ptr->right);
        temp = ptr->left;
        ptr->left = ptr->right;
        ptr->right = temp;
    }
    else
    {
        return;
    }
}

void print(node *ptr)
{
    if (ptr != NULL)
    {
        print(ptr->left);
        printf("%d ", ptr->data);
        print(ptr->right);
    }
    else
    {
        return;
    }
}

typedef struct Node
{
    int data;
    struct Node *left;
    struct Node *right;
} Node;
```

```
Node *createnode(int data)
{
    Node *ptr = (Node *)malloc(sizeof(Node));
    ptr->data = data;
    ptr->left = NULL;
    ptr->right = NULL;
    return (ptr);
}
int fun(Node *root)
{
    int l = 0, r = 0, h = 0, x = 0, y = 0;
    if (root == NULL)
    {
        return 1;
    }
    else if (root->left == NULL && root->right == NULL)
    {
        printf("When node is equal to %d => %d\n", root->data, 1);
        return 1;
    }

    h = root->data;
    if (root->left != NULL)
    {
        l = root->left->data;
    }
    if (root->right != NULL)
    {
        r = root->right->data;
    }
    x = fun(root->left);
    y = fun(root->right);

    if (root->data == (l + r) && x && y)
    {
        printf("When node is equal to %d => %d\n", root->data, 1);
    }
    else
    {
        printf("When node is equal to %d => %d\n", root->data, 0);
        return 0;
    }
}

int main()
{
    printf("\nQuestion-1\n");
    node *ptr = newnode(8);
    ptr->left = newnode(3);
    ptr->right = newnode(10);
    ptr->left->left = newnode(1);
    ptr->left->right = newnode(6);
    ptr->right->right = newnode(14);
    ptr->left->right->left = newnode(4);
    ptr->right->right->left = newnode(13);
    ptr->right->right->right = newnode(7);
```

```
    printf("The Original binary tree is\n");
    print(ptr);
    mirror(ptr);
    printf("\nThe mirror Image of the original binary tree is\n");
    print(ptr);
    printf("\n");

    printf("\nQuestion-2\n");
    printf("\nPart-a\n");
    Node *root = createnode(13);
    root->left = createnode(8);
    root->right = createnode(5);
    root->left->left = createnode(2);
    root->left->right = createnode(6);
    root->right->left = createnode(5);
    fun(root);
    printf("\n\n");
    printf("\nPart-b\n");
    Node *ptr1 = createnode(2);
    ptr1->left = createnode(0);
    ptr1->right = createnode(2);
    ptr1->left->left = createnode(1);
    ptr1->left->right = createnode(0);
    ptr1->right->left = createnode(2);
    ptr1->right->right = createnode(3);
    fun(ptr1);
    printf("\n");

    return 0;
}
```