DSA LAB

PROJECT III B

REGISTRATION NO
BSE-2022-076

PREPARED BY
ZAINAB

HOTEL MANAGEMENT SYSTEM

PROBLEM

"Tree Hotel Manager" is a streamlined hotel management system implementing a binary tree structure for efficient room record management. This system allows users to seamlessly insert, search, update, and delete room records. The use of a binary tree ensures quick access and organized display of room records. Additional features include sorting records by Room ID and searching by Customer Name or Allocated Date, making it a powerful and user-friendly tool for hotel administrators.

CODE

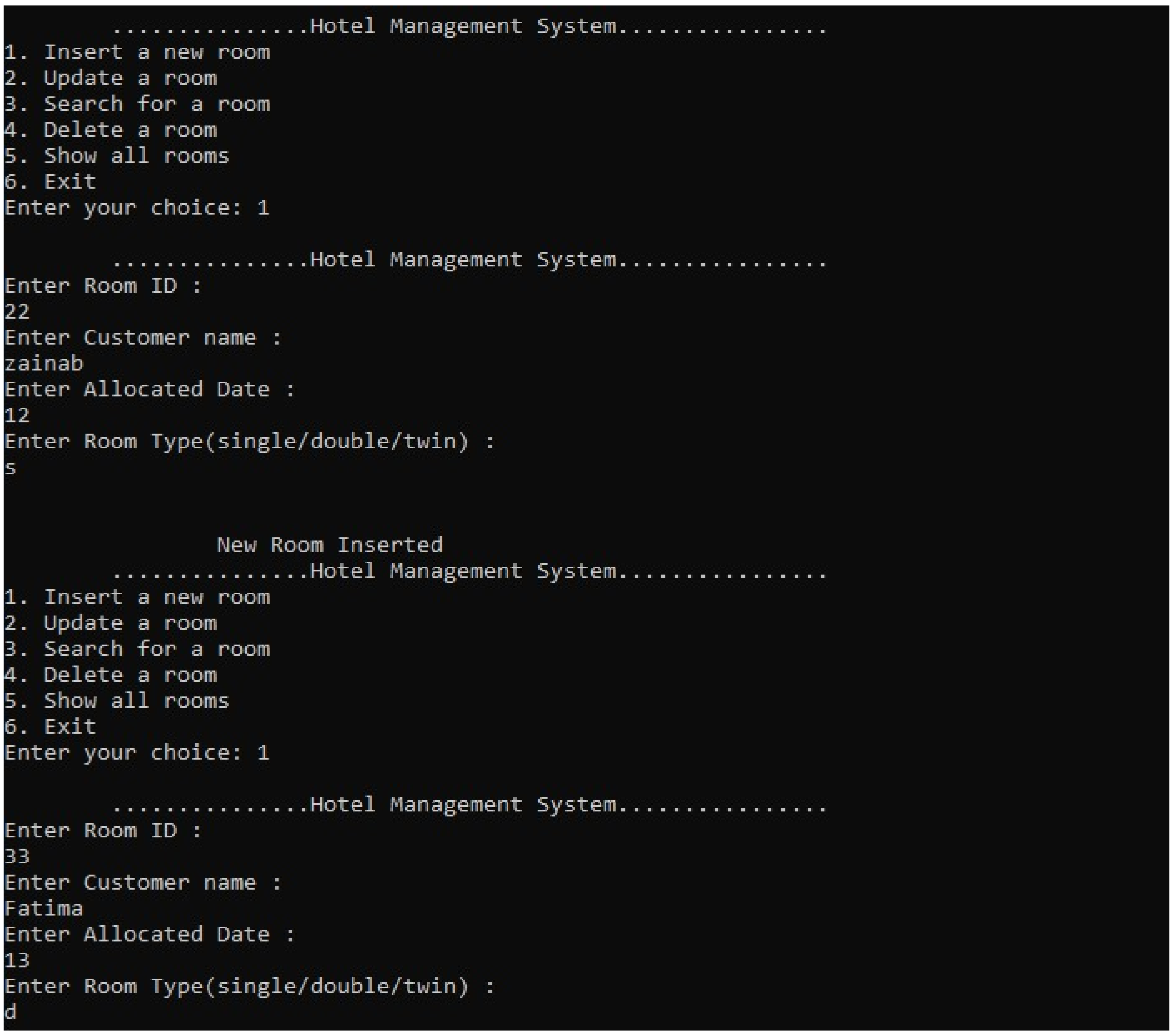
```
#include <iostream>

#include <fstream>

#include <conio.h>

using namespace std;

class TreeNode

{

public:

   int id, date;

   string name, roomtype;

   TreeNode *left, *right;


   TreeNode() : left(NULL), right(NULL)

   {  }
```

```
};

class Hotel
{
public:
    TreeNode *root;

    Hotel() : root(NULL)
    {   };

    void insert();
    void menu();
    void update();
    void search();
    void Delete();
    void show();
    void searchByName();
    void searchByDate();

private:
    TreeNode* insert(TreeNode* root, int id, const string& name, int date, const string& roomtype);
    void inOrderTraversal(TreeNode* root);
    TreeNode* search(TreeNode* root, int id);
    TreeNode* deleteNode(TreeNode* root, int id);
    TreeNode* findMin(TreeNode* root);
    TreeNode* updateNode(TreeNode* root, int id, const string& newName, int newDate, const string& 
newRoomType);
};
TreeNode* Hotel::insert(TreeNode* root, int id, const string& name, int date, const string& roomtype)
{
    if (root == NULL)
    {
        TreeNode *temp = new TreeNode;
        temp->id = id;
        temp->name = name;
        temp->date = date;
        temp->roomtype = roomtype;
        return temp;
    }

    if (id < root->id)
    {
        root->left = insert(root->left, id, name, date, roomtype);
    }
    else if (id > root->id)
    {
        root->right = insert(root->right, id, name, date, roomtype);
    }

    return root;
}

void Hotel::insert()
{
    cout << "\n\t..............Hotel Management System...............";
    int id, date;
    string name, roomtype;

    cout << "\nEnter Room ID :" << endl;
    cin >> id;
    cout << "Enter Customer name :" << endl;
    cin >> name;
    cout << "Enter Allocated Date :" << endl;
    cin >> date;
    cout << "Enter Room Type(single/double/twin) :" << endl;
    cin >> roomtype;

    root = insert(root, id, name, date, roomtype);

    cout << "\n\n\t\tNew Room Inserted";
    getch();
}

TreeNode* Hotel::search(TreeNode* root, int id)
{
    if (root == NULL || root->id == id)
    {
        return root;
    }

    if (id < root->id)
    {
        return search(root->left, id);
    }
    else
    {
        return search(root->right, id);
    }
}

void Hotel::search()
{
    cout << "\n\t..............Hotel Management System...............";
    int t_id;
    if (root == NULL)
    {
        cout << "\n\nBinary tree is Empty";
    }
    else
    {
        cout << "\n\nRoom ID";
        cin >> t_id;
        TreeNode *result = search(root, t_id);

        if (result != NULL)
        {
            cout << "\n\nRoom ID :" << result->id;
            cout << "\n\nCustomer Name :" << result->name;
            cout << "\n\nRoom Allocated Date :" << result->date;
            cout << "\n\nRoom Type :" << result->roomtype;
        }
        else
        {
            cout << "\n\nRoom not found.";
        }
    }
    getch();
}

TreeNode* Hotel::updateNode(TreeNode* root, int id, const string& newName, int newDate, const
string& newRoomType)
{
    if (root == NULL)
    {
        return root;
    }

    if (id < root->id)
    {
        root->left = updateNode(root->left, id, newName, newDate, newRoomType);
    }
    else if (id > root->id)
    {
        root->right = updateNode(root->right, id, newName, newDate, newRoomType);
    }
    else
    {
        root->name = newName;
        root->date = newDate;
        root->roomtype = newRoomType;
    }

    return root;
}

void Hotel::update()
{
    cout << "\n\t..............Hotel Management System...............";
    int t_id;
    if (root == NULL)
    {
        cout << "\n\nBinary tree is Empty";
    }
    else
    {
        cout << "\n\nRoom ID to Update";
        cin >> t_id;
        root = updateNode(root, t_id, "", 0, ""); // You may add prompts for new values if needed
        cout << "\n\n\t\tUpdate Record Successfully";
    }
    getch();
}

TreeNode* Hotel::deleteNode(TreeNode* root, int id)
{
    if (root == NULL)
    {
        return root;
    }

    if (id < root->id)
    {
        root->left = deleteNode(root->left, id);
    }
    else if (id > root->id)
    {
        root->right = deleteNode(root->right, id);
    }
    else
    {
        if (root->left == NULL)
        {
            TreeNode* temp = root->right;
            delete root;
            return temp;
        }
        else if (root->right == NULL)
        {
            TreeNode* temp = root->left;
            delete root;
            return temp;
        }

        TreeNode* temp = findMin(root->right);
        root->id = temp->id;
        root->right = deleteNode(root->right, temp->id);
    }

    return root;
}

TreeNode* Hotel::findMin(TreeNode* root)
{
    while (root->left != NULL)
    {
        root = root->left;
    }
    return root;
}
```

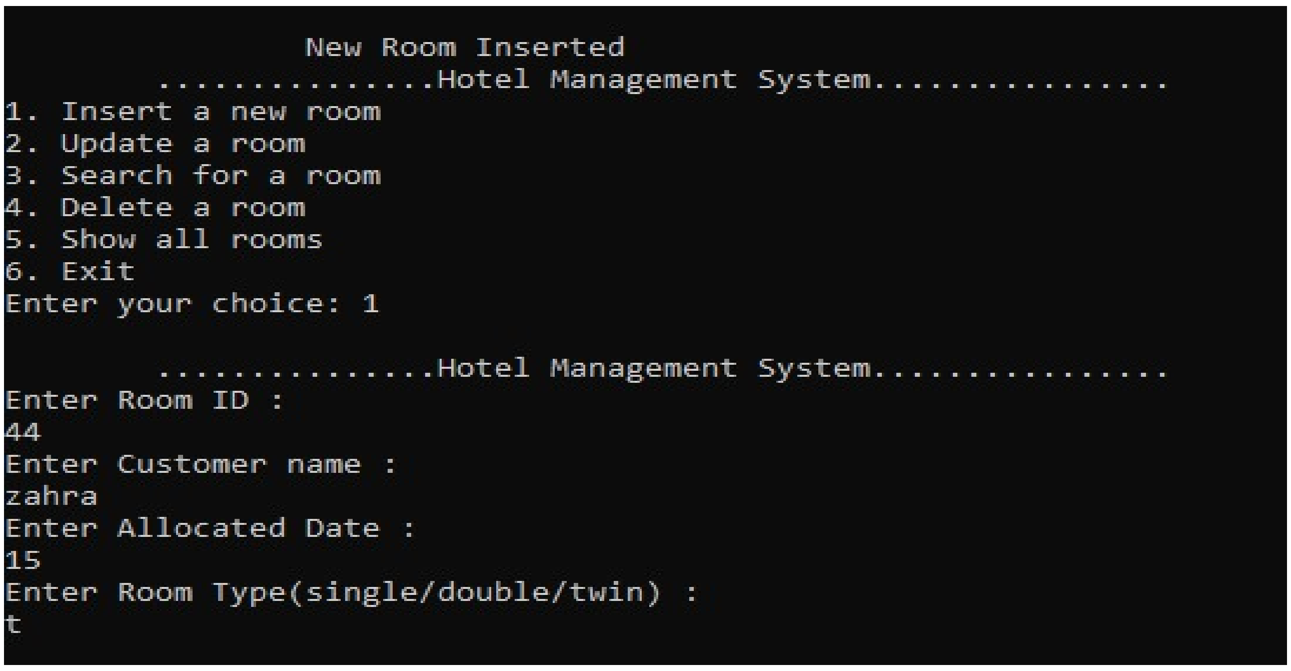
```
void Hotel::Delete()
{
    cout << "\n\t..............Hotel Management System...............";
    int t_id;
    if (root == NULL)
    {
        cout << "\n\nBinary tree is Empty";
    }
    else
    {
        cout << "\n\nRoom ID";
        cin >> t_id;
        root = deleteNode(root, t_id);
        cout << "Delete Room Record Successfully\n";
    }
    getch();
}

void Hotel::inOrderTraversal(TreeNode* root)
{
    if (root != NULL)
    {
        inOrderTraversal(root->left);
        cout << "\n\nRoom ID: " << root->id;
        cout << "\nCustomer Name: " << root->name;
        cout << "\nRoom Allocated Date: " << root->date;
        cout << "\nRoom Type: " << root->roomtype;
        inOrderTraversal(root->right);
    }
}

void Hotel::show()
{
    if (root == NULL)
    {
        cout << "\n\nNo Records Found\n";
    }
    else
    {
        cout << "\n\nRoom Records:\n";
        inOrderTraversal(root);
    }
    getch();
}
void Hotel::menu()
{
    int choice;
    do
    {
        cout << "\n\t..............Hotel Management System...............";
        cout << "\n1. Insert a new room";
        cout << "\n2. Update a room";
        cout << "\n3. Search for a room";
        cout << "\n4. Delete a room";
        cout << "\n5. Show all rooms";
        cout << "\n6. Exit";
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            insert();
            break;
        case 2:
            update();
            break;
        case 3:
            search();
            break;
        case 4:
            Delete();
            break;
        case 5:
            show();
            break;
        case 6:
            cout << "\nExiting the program";
            break;
        default:
            cout << "\nInvalid choice, please try again";
        }

    } while (choice != 6);
}

int main()
{
    Hotel h1;
    h1.menu();
    return 0;
}
```

OUTPUT

Insert a Room

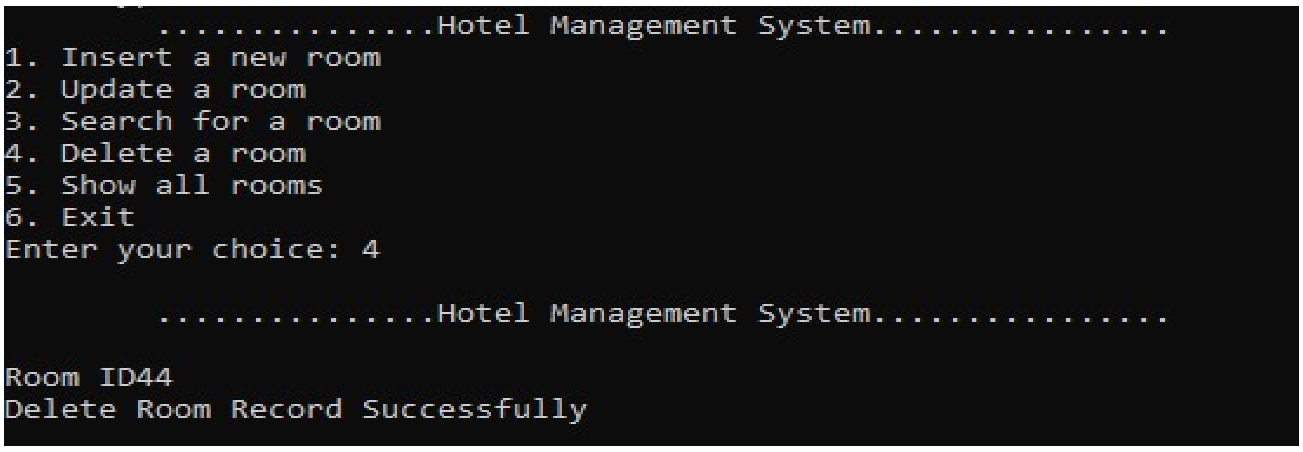


```
        ..............Hotel Management System...............
1. Insert a new room
2. Update a room
3. Search for a room
4. Delete a room
5. Show all rooms
6. Exit
Enter your choice: 1

        ..............Hotel Management System...............
Enter Room ID :
22
Enter Customer name :
zainab
Enter Allocated Date :
12
Enter Room Type(single/double/twin) :
s


                New Room Inserted
        ..............Hotel Management System...............
1. Insert a new room
2. Update a room
3. Search for a room
4. Delete a room
5. Show all rooms
6. Exit
Enter your choice: 1

        ..............Hotel Management System...............
Enter Room ID :
33
Enter Customer name :
Fatima
Enter Allocated Date :
13
Enter Room Type(single/double/twin) :
d
```



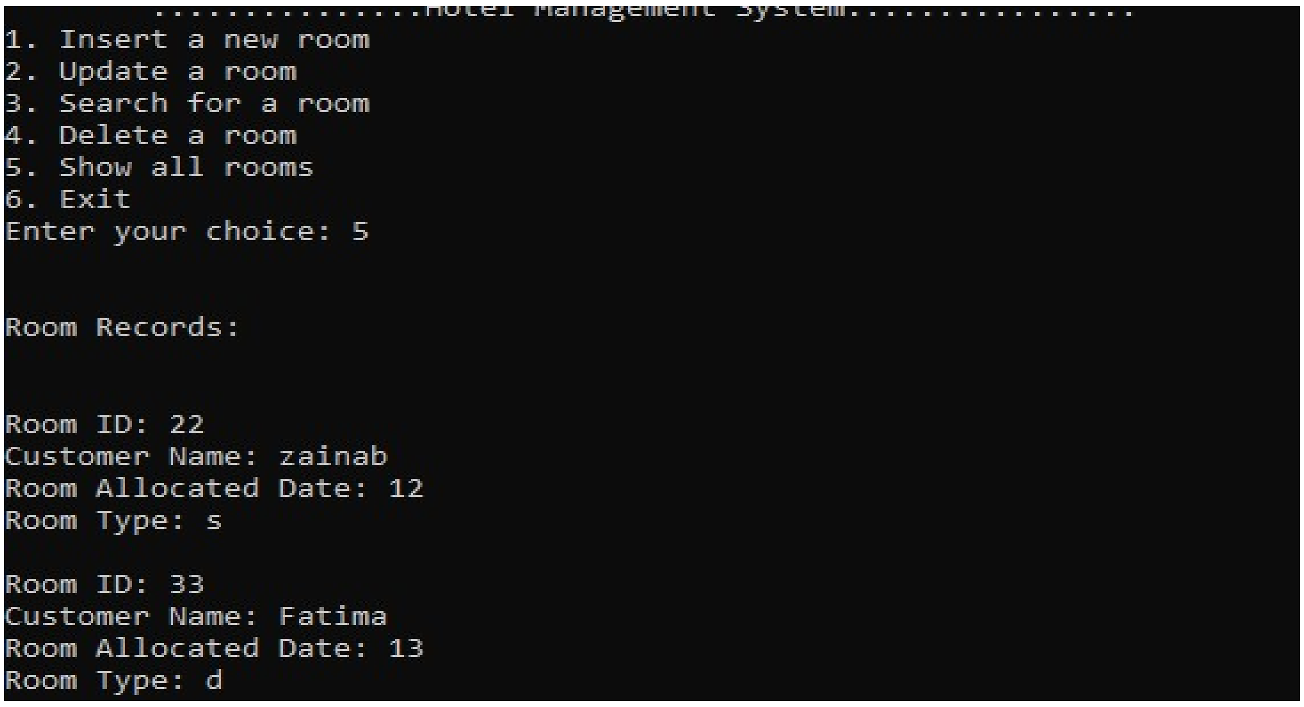
```
                New Room Inserted
        ...............Hotel Management System................
1. Insert a new room
2. Update a room
3. Search for a room
4. Delete a room
5. Show all rooms
6. Exit
Enter your choice: 1

        ...............Hotel Management System................
Enter Room ID :
44
Enter Customer name :
zahra
Enter Allocated Date :
15
Enter Room Type(single/double/twin) :
t
```

Deleting 44 Room Id



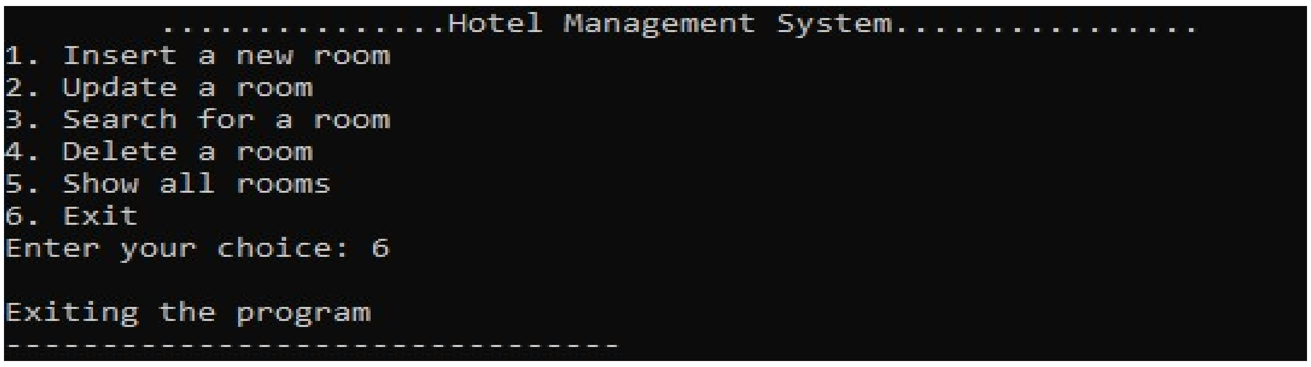
```
        ...............Hotel Management System................
1. Insert a new room
2. Update a room
3. Search for a room
4. Delete a room
5. Show all rooms
6. Exit
Enter your choice: 4

        ...............Hotel Management System................

Room ID44
Delete Room Record Successfully
```

Showing ROOM after deleting room id 44



```
        ...............Hotel Management System................
1. Insert a new room
2. Update a room
3. Search for a room
4. Delete a room
5. Show all rooms
6. Exit
Enter your choice: 5


Room Records:


Room ID: 22
Customer Name: zainab
Room Allocated Date: 12
Room Type: s

Room ID: 33
Customer Name: Fatima
Room Allocated Date: 13
Room Type: d
```

Enter 6 to exit



```
        ...............Hotel Management System................
1. Insert a new room
2. Update a room
3. Search for a room
4. Delete a room
5. Show all rooms
6. Exit
Enter your choice: 6

Exiting the program
--------------------------------
```