IIT 2025105

Assignment - 8

Q1

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

Struct Node {
    int id;
    struct Node *left;
    struct Node *right;
};

struct Node * CreateNode (int id)
{
    struct Node *newNode = (struct Node *) malloc(
                           Size of(struct Node));
    newNode -> id = id;
    newNode -> left = NULL;
    newNode -> right = NULL;
    return newNode;
}

Struct Node *insert (struct Node * root, int id)
    {
        if (root == NULL) {
                return CreateNode (id);
        }

        if (id < root -> id) {
                root -> left = insert (root -> left, id);
```

```
else if (id > root->id) {
        root->right = insert(root->right, id);
    }
    return root;
}

int search(struct Node *root, int id) {
    if (root == NULL) {
        return 0;
    }
    if (root->id == id) {
        return 1;
    }
    if (id < root->id) {
        return search(root->left, id);
    }
    else {
        return search(root->right, id);
    }
}

Void inorder(struct Node *root) {
    if (root == NULL) {
        return;
    }
    inorder(root->left);
    printf("%d", root->id);
    inorder(root->right);
}
```

```
int main() {
    printf("Enter no. of operations:-\n");
    int N;
    scanf("%d", &N);

    Struct Node * root = NULL;
    Char operation[20];
    int value;

    for (int i=0 ; i<N ; i++) {
        scanf("%s", operation);

        if (strcmp(operation, "ADD") == 0) {
            scanf("%d", &value);
            root = insert(root, value);
        }
        else if (strcmp(operation, "SEARCH") == 0) {
            scanf("%d", &value);
            if (search(root, value)) {
                printf("FOUND \n");
            }
            else {
                printf("NOT FOUND \n");
            }
        }

        else if (strcmp(operation, "INORDER") == 0) {
            if (root = NULL) {
                printf("EMPTY");
            }
```

```
    else {
        inorder(root);
        printf("\n");
    }
  }
}
return 0;
}
```

02

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

struct Node {
    int id;
    struct Node* left;
    struct Node* right;
};

struct Node* createNode(int id) {

    struct Node* newNode = (struct Node*) malloc
                (sizeof(struct Node));
    newNode->id = id;
    newNode->left = NULL;
    newNode->right = NULL;

}
```

```
void inorder ( struct node * root) {
    if (root = NULL) {
                return;
            }

    inOrder (root -> left);
    printf (" %.d ", root -> id);
    inorder( root -> right);
}

Struct Node * findmin (struct Node * root) {
    while (root -> left != NULL) {
            root = root -> left;
    } return root;
}

Struct Node * deleteNode (struct Node * root, int id, int
                          * found ) {
    if (root == NULL)
                return NULL;

    if (id < root -> id) {
        root -> left = deleteNode (root -> left, id, found);
    }
    else if (id > root -> id) {
        root -> right = delete Node , id, found);

    }
```

```
else {
        * found = 1;

    if (root -> left == NULL && root -> right == NULL)
    {
        free (root);
        return NULL;
    }
    else if( root -> left == NULL) {

        struct Node * temp = root -> right;
        free(root);
        return temp;
    }

    else {

        struct Node * temp = FindMin( root -> right);
        root -> id = temp -> id;
        root -> right = deleteNode (root -> right,
                                    temp -> id, found);

    }
}
}

int main() {
printf (" Enter No. of operations :- \n");
int N;
scanf( " %d", &N);
```

```
struct Node* root = NULL;
char operation[20];
int value;

for(int i=0; i<N; i++){

    scanf("%s", operation);

    if(strcmp(operation, "REGISTER")==0){

        scanf("%d", &value);
        root = insert(root, value);
    }
    else if(strcmp(operation, "DISCHARGE")==0){

        scanf("%d", &value);
        int found=0;
        root = deleteNode(root, value, &found);

        if(!found){
            printf("NOT FOUND\n");
        }
    else if(strcmp(operation, "LIST")==0){

        if(root==NULL){
            printf("EMPTY\n");
        }
        else
            {
                inorder(root);
                printf("\n");
            }
    }
    return 0;
}
```