The University of Faisalabad

ASSIGNMENT NUMBER 3

SUBMITTED BY : M. WAHAAJ SIDDIQUE

SUBMITTED TO : MAM IRSHA QURAISHI

REG # : 2023-BS-AI-007

DEPARTMENT : COMPUTER SCIENCES (AI-(SEC-A))

DSA Assignment

Doubly Linked List: Write a program to delete the first node in a doubly linked list.

```
void deleteFirst() {
    if (!head) return;
    DNode* temp = head;
    head = head->next;
    if (head) head->prev = nullptr;
    delete temp;
}
```

How can you delete the last node in a doubly linked list? Write the code.

```
void deleteLast() {
    if (!head) return;
    if (!head->next) {
        delete head;
        head = nullptr;
        return;
    }
    DNode* temp = head;
    while (temp->next) temp = temp->next;
    temp->prev->next = nullptr;
    delete temp;
}
```

Write code to delete a node by its value in a doubly linked list.

```
void deleteByValue(int value) {
    if (!head) return;
    if (head->data == value) {
        deleteFirst();
        return;
    }
    DNode* temp = head;
    while (temp && temp->data != value) temp = temp->next;
    if (!temp) return;
    if (temp->next) temp->next->prev = temp->prev;
    if (temp->prev) temp->prev->next = temp->next;
    delete temp;
}
```

How would you delete a node at a specific position in a doubly linked list? Show it in code.

```
void deleteAtPosition(int pos) {
    if (!head || pos < 1) return;
    if (pos == 1) {
        deleteFirst();
        return;
    }
    DNode* temp = head;
    for (int i = 1; temp && i < pos; ++i) temp = temp->next;
    if (!temp) return;
    if (temp->next) temp->next->prev = temp->prev;
    if (temp->prev) temp->prev->next = temp->next;
    delete temp;
}
```

After deleting a node, how will you write the forward and reverse traversal functions?

```
void traverseForward() {
    DNode* temp = head;
    while (temp) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

void traverseReverse() {
    if (!head) return;
    DNode* temp = head;
    while (temp->next) temp = temp->next;
    while (temp) {
        cout << temp->data << " ";
        temp = temp->prev;
    }
    cout << endl;
}
```

Circular Linked List: Write a program to delete the first node in a circular linked list.

```
void deleteFirst() {
    if (!head) return;
    if (head->next == head) {
        delete head;
        head = nullptr;
        return;
    }
    CNode* temp = head;
    CNode* last = head;
    while (last->next != head) last = last->next;
    head = head->next;
    last->next = head;
    delete temp;
}
```

How can you delete the last node in a circular linked list? Write the code.

```
void deleteLast() {
    if (!head) return;
    if (head->next == head) {
        delete head;
        head = nullptr;
        return;
    }
    CNode* temp = head;
    while (temp->next->next != head) temp = temp->next;
    delete temp->next;
    temp->next = head;
}
```

Write a function to delete a node by its value in a circular linked list.

```
void deleteByValue(int value) {
    if (!head) return;
    if (head->data == value) {
        deleteFirst();
        return;
    }
    CNode* temp = head;
    while (temp->next != head && temp->next->data != value) temp = temp->next;
    if (temp->next->data == value) {
        CNode* toDelete = temp->next;
        temp->next = temp->next->next;
        delete toDelete;
    }
}
```

How will you delete a node at a specific position in a circular linked list? Write code for it.

```
void deleteAtPosition(int pos) {
    if (!head || pos < 1) return;
    if (pos == 1) {
        deleteFirst();
        return;
    }
    CNode* temp = head;
    for (int i = 1; temp->next != head && i < pos - 1; ++i) temp = temp->next;
    if (temp->next != head) {
        CNode* toDelete = temp->next;
        temp->next = temp->next->next;
        delete toDelete;
    }
}
```

Write a program to show forward traversal after deleting a node in a circular linked list.

```
void traverseForward() {
    if (!head) return;
    CNode* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;
}
```

Binary Search Tree: Write a program to count all the nodes in a binary search tree.

```
int countNodes(BSTNode* node) {
    if (!node) return 0;
    return 1 + countNodes(node->left) + countNodes(node->right);
}
```

How can you search for a specific value in a binary search tree? Write the code.

```
bool search(int value) {
    return searchRec(root, value);
}

bool searchRec(BSTNode* node, int value) {
    if (!node) return false;
    if (node->data == value) return true;
    if (value < node->data) return searchRec(node->left, value);
    return searchRec(node->right, value);
}
```

Write code to traverse a binary search tree in in-order, pre-order, and post-order.

```
void inOrderTraversal(BSTNode* node) {
    if (!node) return;
    inOrderTraversal(node->left);
    cout << node->data << " ";
    inOrderTraversal(node->right);
}

void preOrderTraversal(BSTNode* node) {
    if (!node) return;
    cout << node->data << " ";
    preOrderTraversal(node->left);
    preOrderTraversal(node->right);
}

void postOrderTraversal(BSTNode* node) {
    if (!node) return;
    postOrderTraversal(node->left);
    postOrderTraversal(node->right);
    cout << node->data << " ";
}
```

How will you write reverse in-order traversal for a binary search tree? Show it in code.

```
void reverseInOrderTraversal(BSTNode* node) {
    if (!node) return;
    reverseInOrderTraversal(node->right);
    cout << node->data << " ";
    reverseInOrderTraversal(node->left);
}
```

Write a program to check if there are duplicate values in a binary search tree.

```
bool hasDuplicates(BSTNode* node, set<int>& values) {
    if (!node) return false;
    if (values.count(node->data)) return true;
    values.insert(node->data);
    return hasDuplicates(node->left, values) || hasDuplicates(node->right, values);
}
```

How can you delete a node from a binary search tree? Write code for deleting a leaf, a node with one child, and a node with two children.

```
BSTNode* deleteNodeRec(BSTNode* node, int value) {
    if (!node) return nullptr;
    if (value < node->data) node->left = deleteNodeRec(node->left, value);
    else if (value > node->data) node->right = deleteNodeRec(node->right, value);
    else {
        if (!node->left) {
            BSTNode* temp = node->right;
            delete node;
            return temp;
        } else if (!node->right) {
            BSTNode* temp = node->left;
            delete node;
            return temp;
        }
        BSTNode* temp = minValueNode(node->right);
        node->data = temp->data;
        node->right = deleteNodeRec(node->right, temp->data);
    }
    return node;
}

BSTNode* minValueNode(BSTNode* node) {
    BSTNode* current = node;
    while (current && current->left) current = current->left;
    return current;
}
```