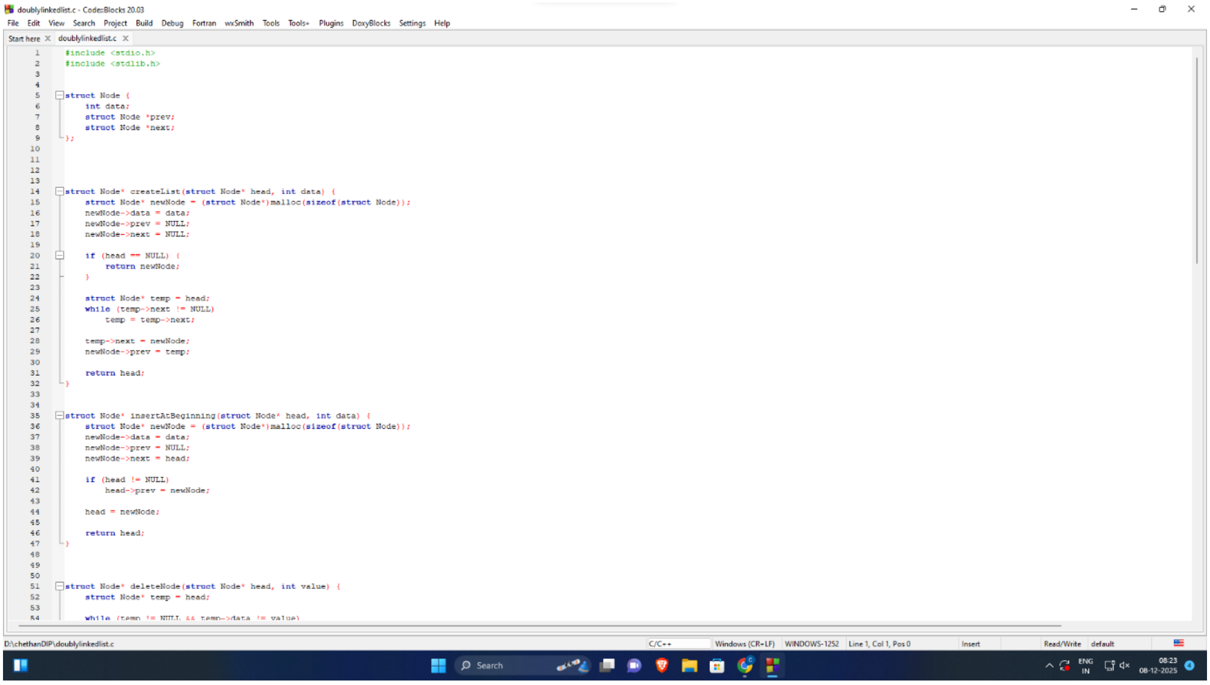
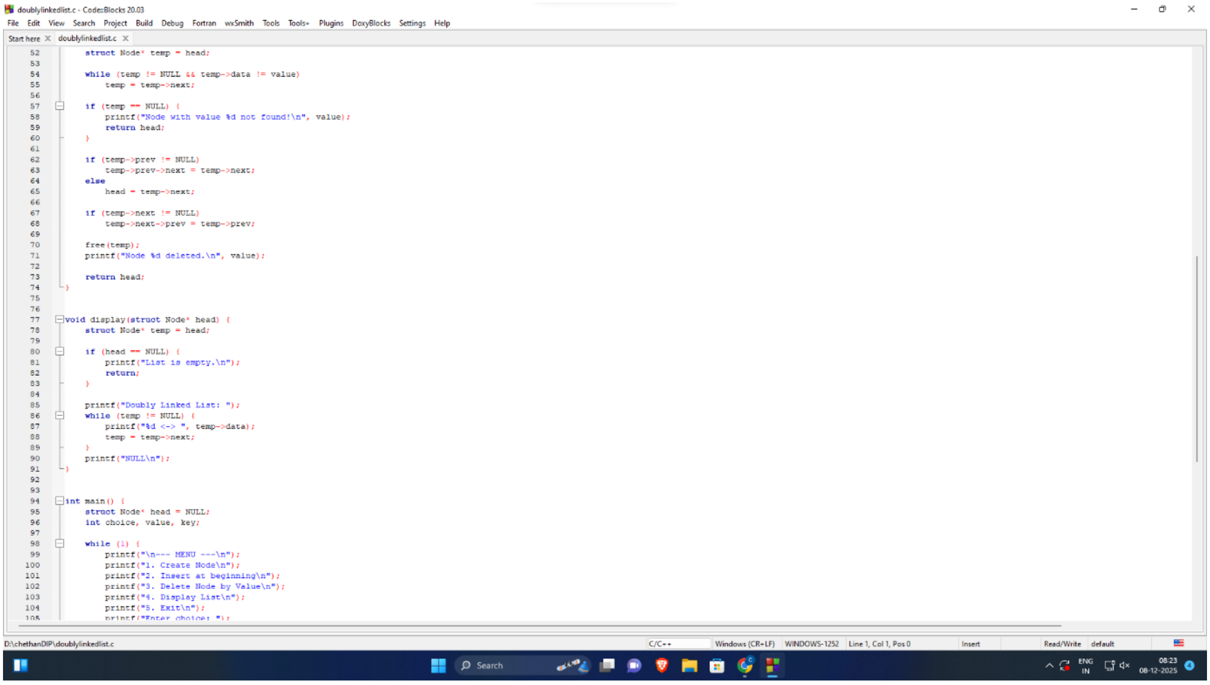
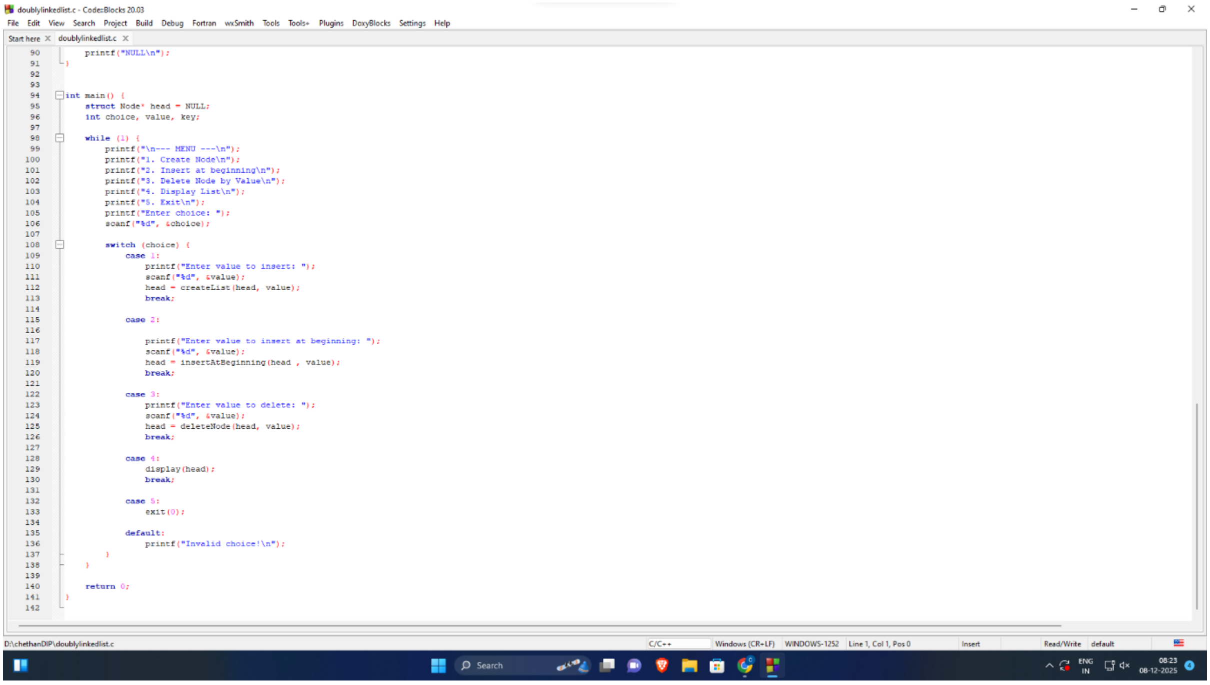

doublylinkedlist.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
doublylinkedlist.c
D:\chethanDIP\doublylinkedlist.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default
Search
ENG IN
08:23
08-12-2025

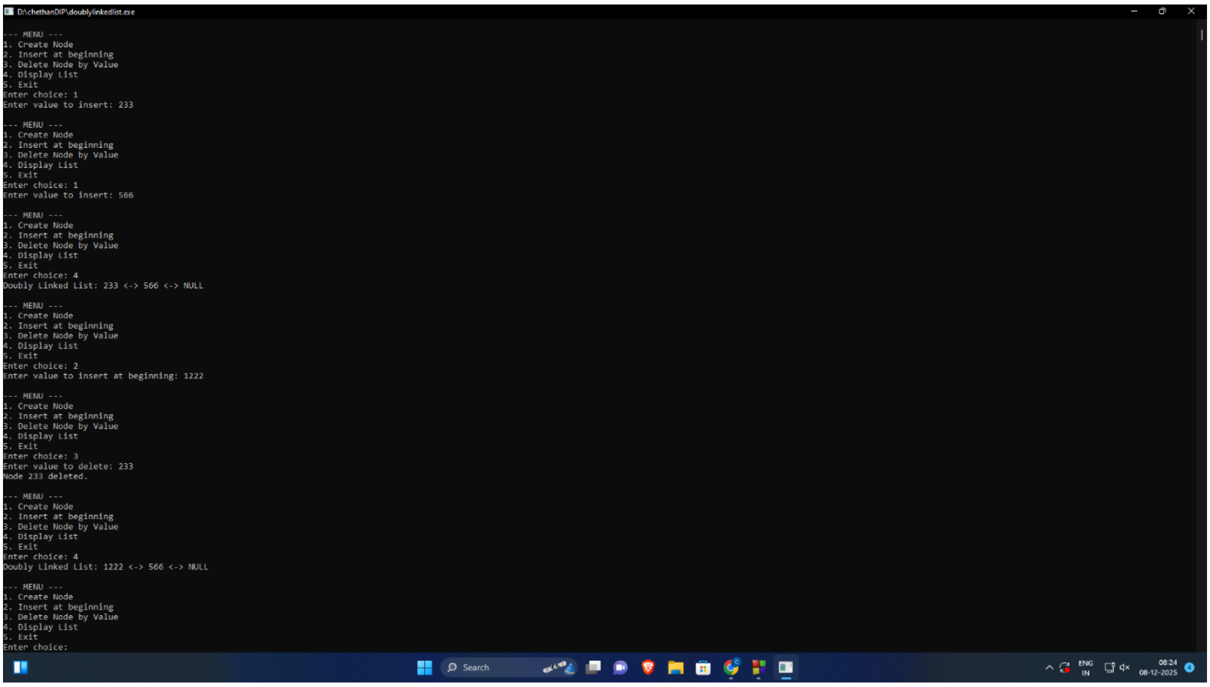
```c
#include <stdio.h>
#include <stdlib.h>


struct Node {
    int data;
    struct Node *prev;
    struct Node *next;
};




struct Node* createList(struct Node* head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;

    if (head == NULL) {
        return newNode;
    }

    struct Node* temp = head;
    while (temp->next != NULL)
        temp = temp->next;

    temp->next = newNode;
    newNode->prev = temp;

    return head;
}


struct Node* insertAtBeginning(struct Node* head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = head;

    if (head != NULL)
        head->prev = newNode;

    head = newNode;

    return head;
}



struct Node* deleteNode(struct Node* head, int value) {
    struct Node* temp = head;

    while (temp != NULL && temp->data != value)
```


doublylinkedlist.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
doublylinkedlist.c
D:\chethanDIP\doublylinkedlist.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default
Search
ENG IN
08:23
08-12-2025

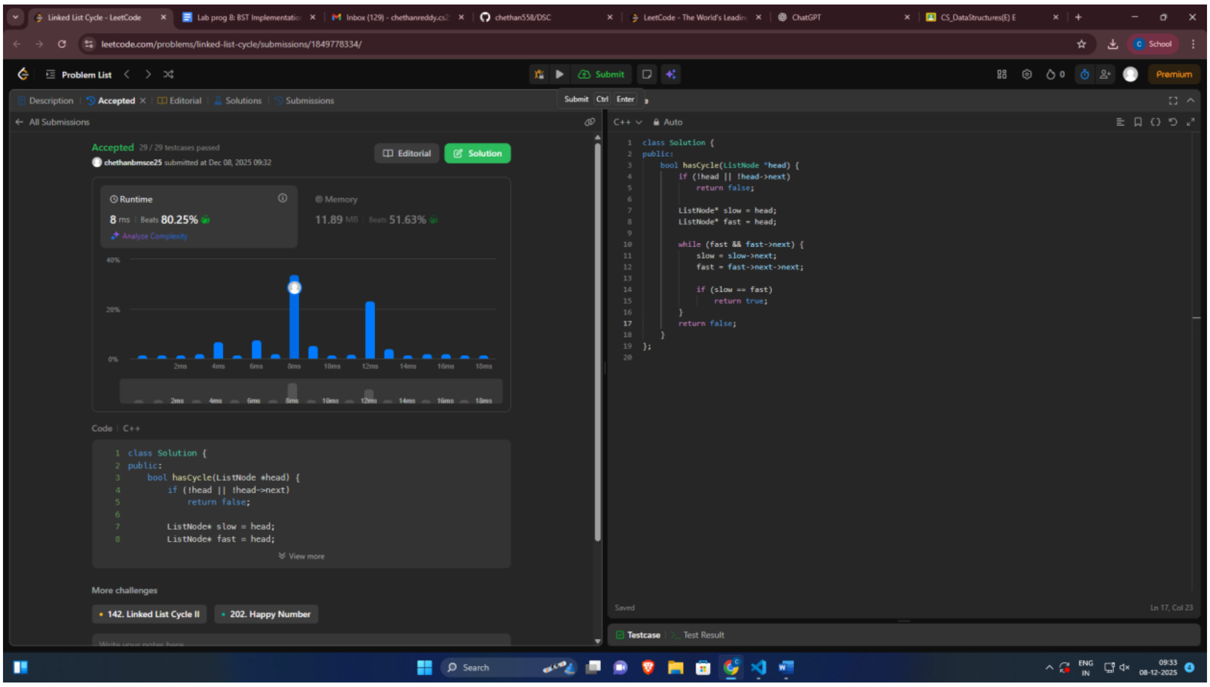
```c
    struct Node* temp = head;

    while (temp != NULL && temp->data != value)
        temp = temp->next;

    if (temp == NULL) {
        printf("Node with value %d not found!\n", value);
        return head;
    }

    if (temp->prev != NULL)
        temp->prev->next = temp->next;
    else
        head = temp->next;

    if (temp->next != NULL)
        temp->next->prev = temp->prev;

    free(temp);
    printf("Node %d deleted.\n", value);

    return head;
}


void display(struct Node* head) {
    struct Node* temp = head;

    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }

    printf("Doubly Linked List: ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}


int main() {
    struct Node* head = NULL;
    int choice, value, key;

    while (1) {
        printf("\n--- MENU ---\n");
        printf("1. Create Node\n");
        printf("2. Insert at beginning\n");
        printf("3. Delete Node by Value\n");
        printf("4. Display List\n");
        printf("5. Exit\n");
        printf("Enter choice: ");
```


doublylinkedlist.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
doublylinkedlist.c
D:\chethanDIP\doublylinkedlist.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default
Search
ENG IN
08:23
08-12-2025

```c
    printf("NULL\n");
}


int main() {
    struct Node* head = NULL;
    int choice, value, key;

    while (1) {
        printf("\n--- MENU ---\n");
        printf("1. Create Node\n");
        printf("2. Insert at beginning\n");
        printf("3. Delete Node by Value\n");
        printf("4. Display List\n");
        printf("5. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                head = createList(head, value);
                break;

            case 2:

                printf("Enter value to insert at beginning: ");
                scanf("%d", &value);
                head = insertAtBeginning(head , value);
                break;

            case 3:
                printf("Enter value to delete: ");
                scanf("%d", &value);
                head = deleteNode(head, value);
                break;

            case 4:
                display(head);
                break;

            case 5:
                exit(0);

            default:
                printf("Invalid choice!\n");
        }
    }

    return 0;
}
```



```
--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 1
Enter value to insert: 233

--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 1
Enter value to insert: 566

--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 4
Doubly Linked List: 233 <-> 566 <-> NULL

--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 2
Enter value to insert at beginning: 1222

--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 3
Enter value to delete: 233
Node 233 deleted.

--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 4
Doubly Linked List: 1222 <-> 566 <-> NULL

--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice:
```



leetcode.com/problems/linked-list-cycle/submissions/1849778334/

Accepted 29 / 29 testcases passed

chethanbmsce25 submitted at Dec 08, 2025 09:32

Runtime: 8 ms | Beats 80.25%

Memory: 11.89 MB | Beats 51.63%

Code | C++

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if (!head || !head->next)
            return false;

        ListNode* slow = head;
        ListNode* fast = head;
```

142. Linked List Cycle II

202. Happy Number

Write your notes here

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if (!head || !head->next)
            return false;

        ListNode* slow = head;
        ListNode* fast = head;

        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast)
                return true;
        }
        return false;
    }
};
```

Saved

Ln 17, Col 23

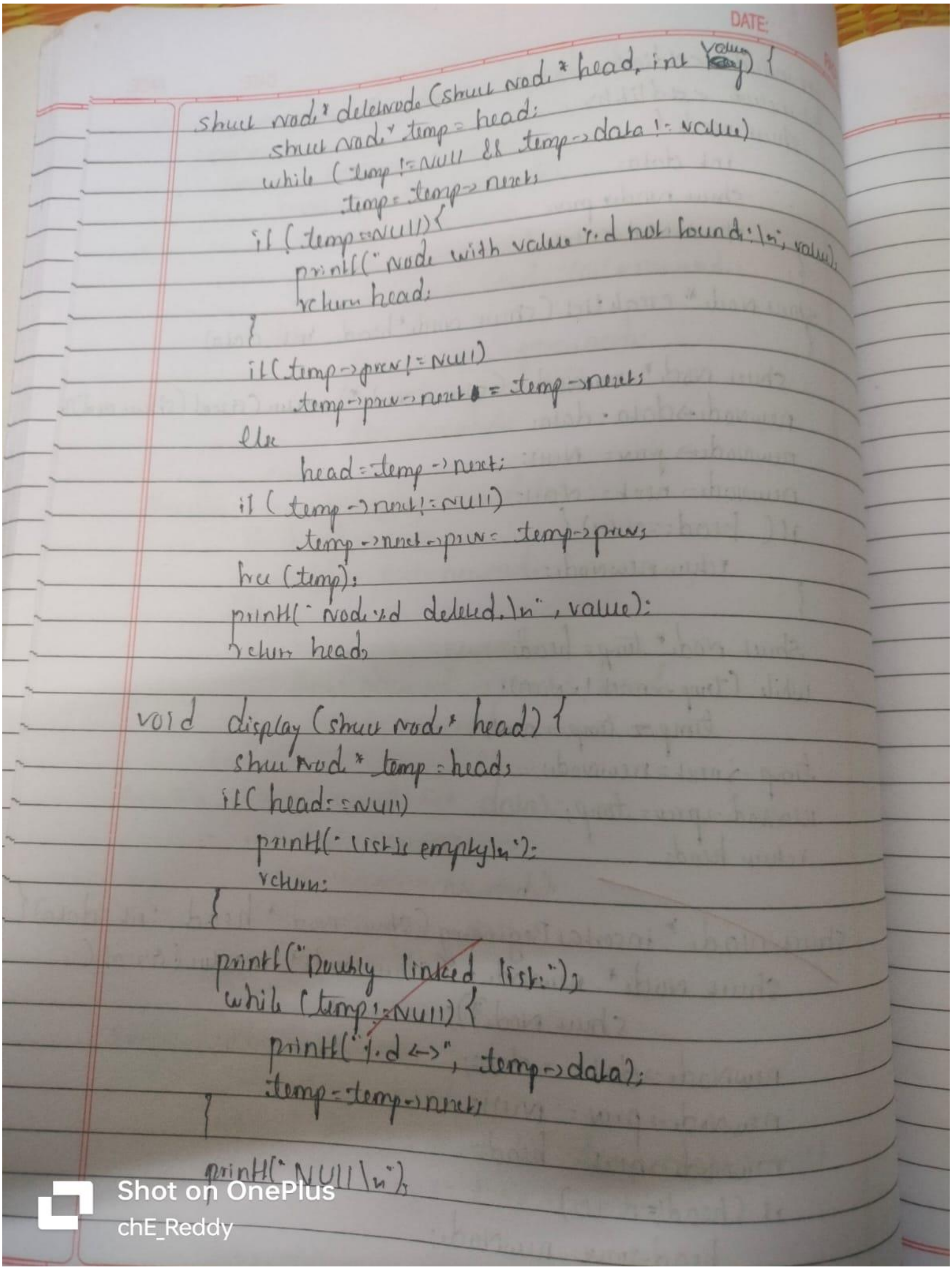
09:33
08-12-2025

7) Implementation of double dinked list

pseudocode

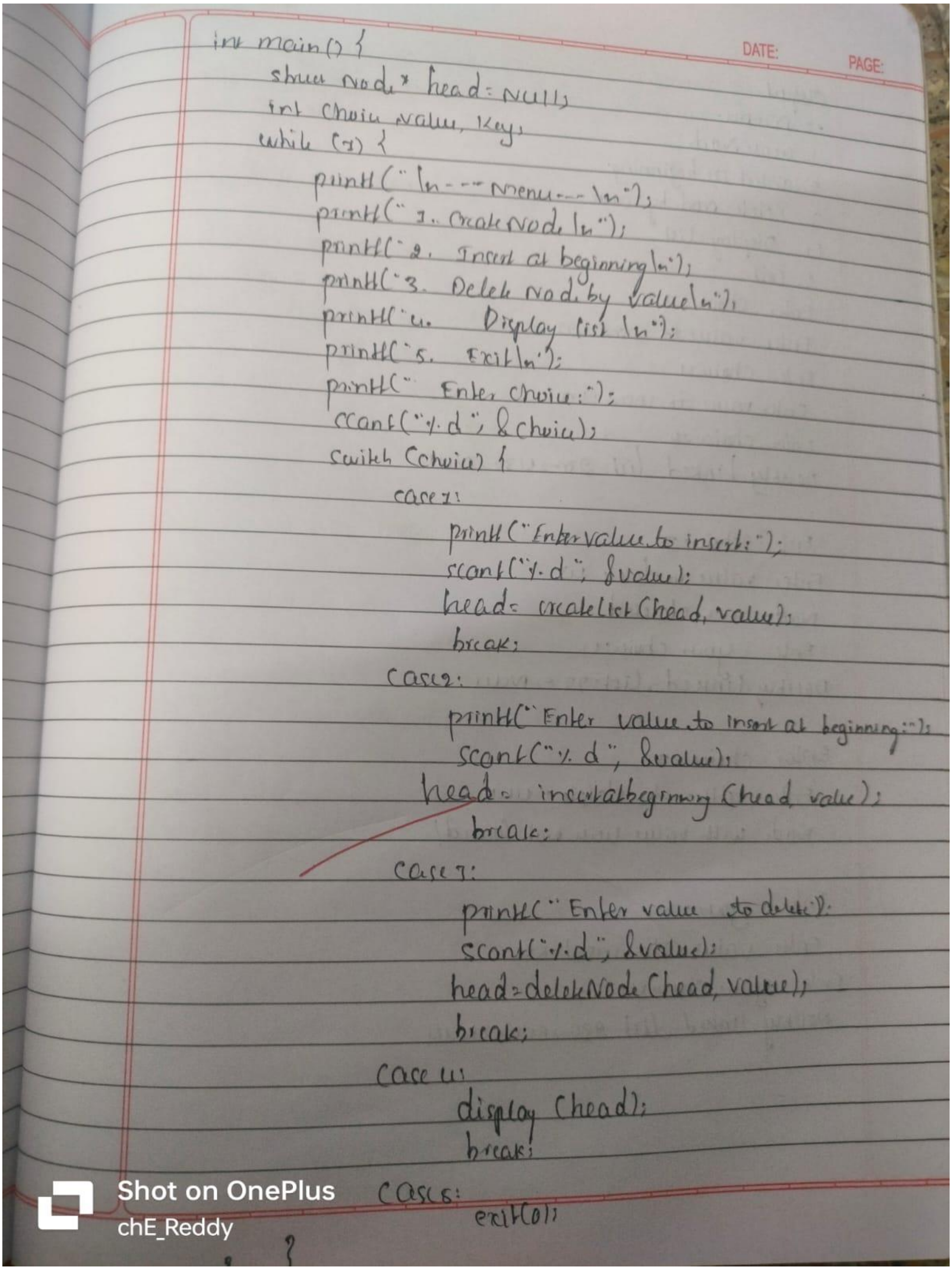
```
Function createnode (int n):
    create a node = (newnode)
      newnode->data = data
      newnode->prev = newnode -> next = NULL
      if(tail == null) head = tail = newnode;
      else
          tail -> next = Newnode;
          newnode -> prev = tail;
          tail = newnode;
      end
Function insert at begging (int data)
    if head is null
        head = tail = newnode;
    else:
        head->prev = newnode.
        ~~head~~ Newnode -> next = head;
        head = new node
    end if
Function insert atend(int data):
     if(tail == NULL)
        tail->next = newnode);
        newnode.prev = tail;
        tail = newnode;
    end if
Function delete (int val):
     Struct temp = head
     while ( temp != null && temp.data != val) {
              temp = temp -> next;
     }
     temp.prev.next = temp.next
     temp.next.prev = temp.prev
```

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
        int data;
        struct node* prev;
        struct Node* next;
};
struct Node* createlist (struct Node* head, int data)
{
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    if( head==NULL) {
          return newNode;
    }
    struct Node* temp = head;
    while (temp->next != NULL).
          temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
    return head;

struct Node* insertatBeginning (struct Node* head, int data){
    struct Node* newNode = (struct Node*)malloc(sizeof(
              struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = head;
    if (head!=NULL)
        head->prev = newNode;
    head = newNode;
    return head;
```



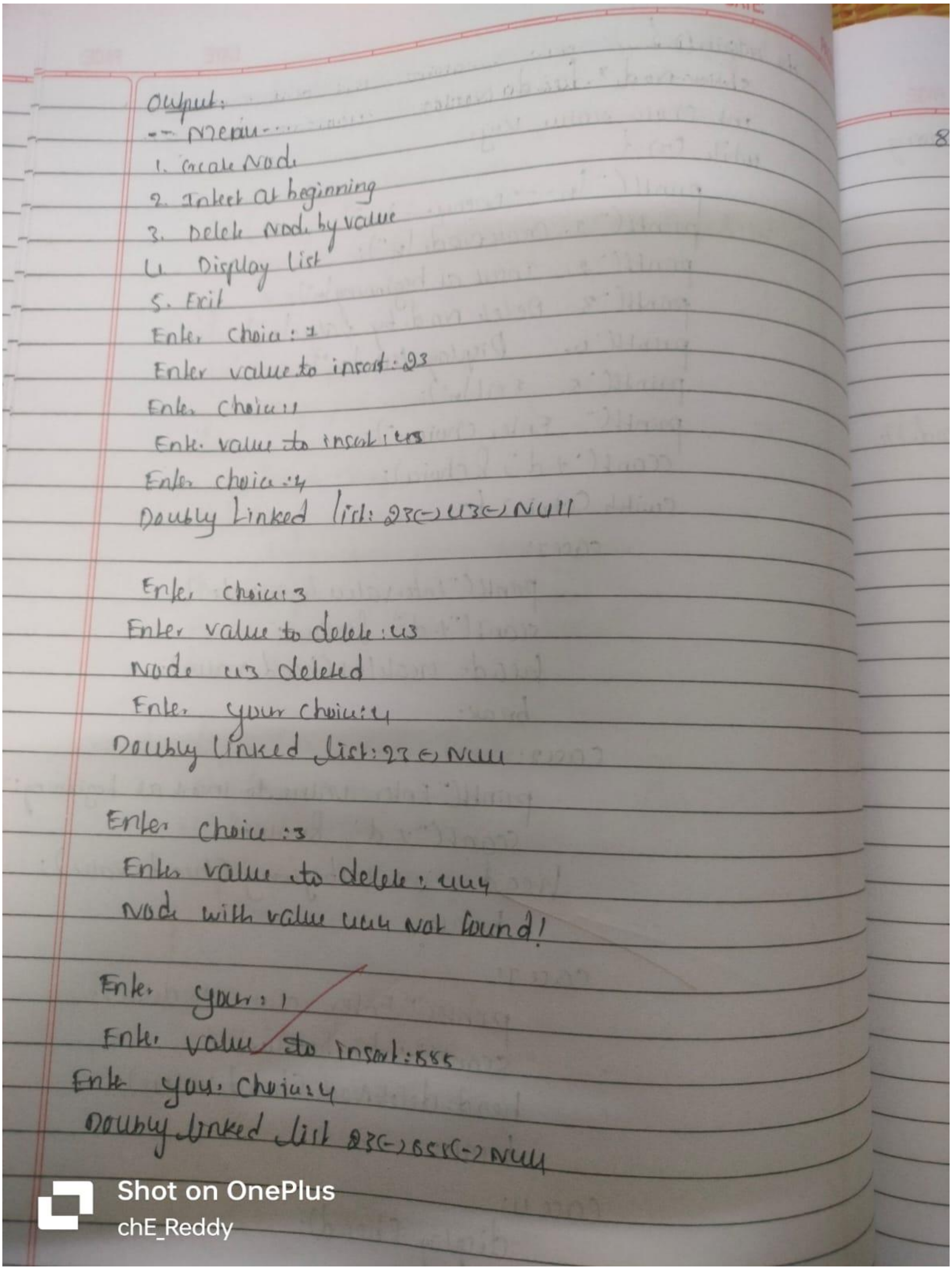
DATE:

```
struct node* deletenode (struct node* head, int ~~key~~ value) {
    struct node* temp = head;
    while (temp != NULL && temp->data != value)
        temp = temp->next;
    if (temp == NULL){
        printf("node with value %d not found.\n", value);
        return head;
    }
    if(temp->prev != NULL)
        temp->prev->next = temp->next;
    else
        head = temp->next;
    if (temp->next != NULL)
        temp->next->prev = temp->prev;
    free (temp);
    printf("node %d deleted.\n", value);
    return head;

void display (struct node* head) {
    struct node* temp = head;
    if(head==NULL)
        printf("list is empty\n");
        return;
    }

    printf("Doubly linked list:");
    while (temp != NULL) {
        printf("%d <->", temp->data);
        temp = temp->next;
    }

    printf("NULL\n");
```



DATE: PAGE:

```
int main() {
    struct node* head=NULL;
    int choice value, key;
    while (1) {
        printf("\n---Menu---\n");
        printf("1. Create Node\n");
        printf("2. Insert at beginning\n");
        printf("3. Delete Node by value\n");
        printf("4.   Display list \n");
        printf("5.  Exit\n");
        printf(" Enter choice:");
        scanf("%d", &choice);
        switch (choice) {
            case1:
                printf("Enter value to insert:");
                scanf("%d", &value);
                head= createlist(head, value);
                break;
            case2:
                printf("Enter value to insert at beginning:");
                scanf("%d", &value);
               head= insertatbeginning(head value);
                break;
            case3:
                printf(" Enter value  to delete").
                scanf("%d", &value);
                head=deleteNode(head, value);
                break;
            case 4:
                display (head);
                break;
            case5:
                exit(0);
    }
```



Output:

```
-- Menu--
1. Create Node
2. Inseet at beginning
3. Delete Node by value
4. Display List
5. Exit
Enter choice: 1
Enter value to insert: 23
Enter choice: 1
Enter value to insert: 43
Enter choice: 4
Doubly Linked list: 23<->43<->NULL

Enter choice: 3
Enter value to delete: 43
Node 43 deleted
Enter your choice: 4
Doubly Linked list: 23<->NULL

Enter choice: 3
Enter value to delete: 444
Node with value 444 Not found!

Enter your: 1
Enter value to insert: 855
Enter your choice: 4
Doubly linked list 23<->855<->NULL
```

DATE: PAGE:

8) ~~To construct a binary search tree and traverse the tree using inorder, preorder, postorder and display elements in the tree.~~

leetcode :- linked list cycle

```
class solution{
public:
        bool hascycle((ListNode* head){
            if(!head || !head->next)
                return fale;
            ListNode*slow= head
            ListNode* fast= head;
           while (fast && fast->next)
              slow= slow->next;
              fast= fast->next->next;
              if (slow==fast)
                  return true;
        }
        return fale;
      }
}
```

A

8/11/25