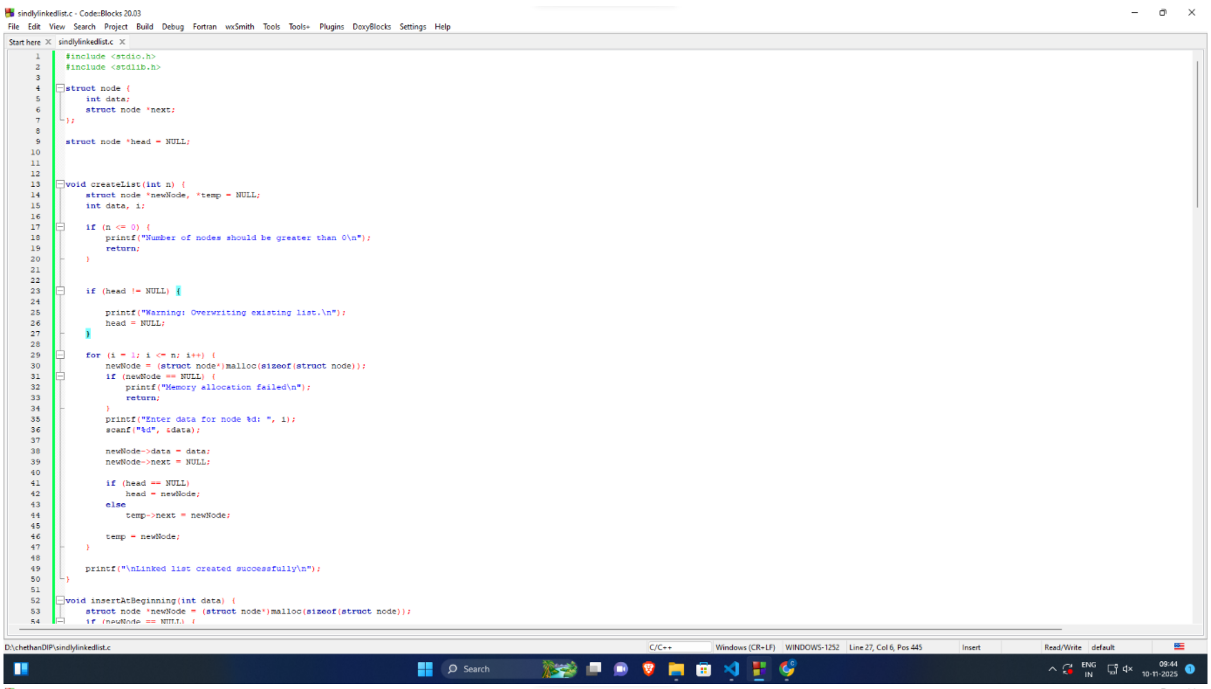
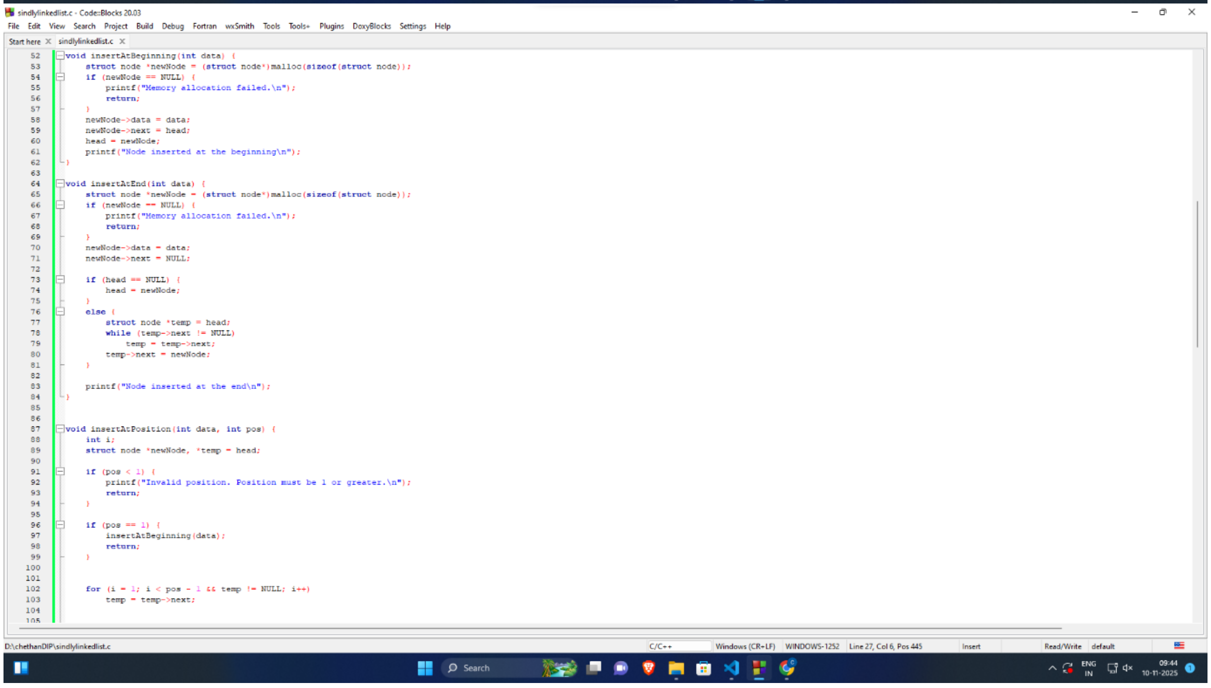

sindlylinkedlist.c - Code::Blocks 20.03

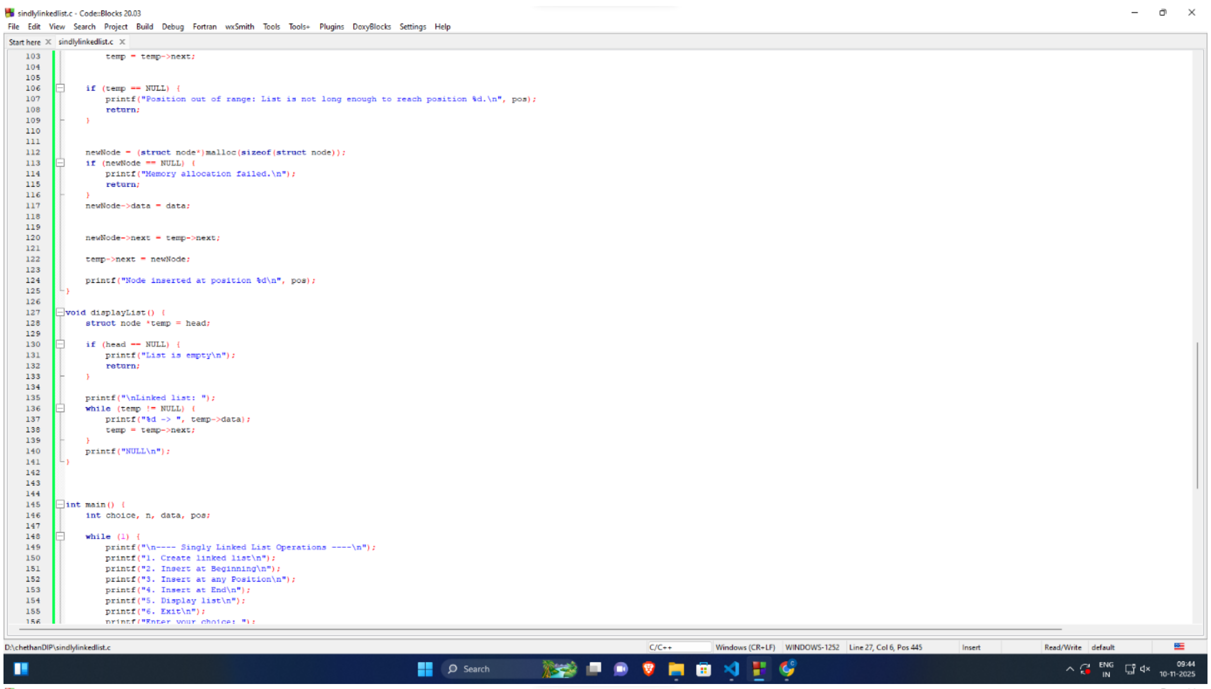
```c
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *next;
};

struct node *head = NULL;



void createList(int n) {
    struct node *newNode, *temp = NULL;
    int data, i;

    if (n <= 0) {
        printf("Number of nodes should be greater than 0\n");
        return;
    }


    if (head != NULL) {

        printf("Warning: Overwriting existing list.\n");
        head = NULL;
    }

    for (i = 1; i <= n; i++) {
        newNode = (struct node*)malloc(sizeof(struct node));
        if (newNode == NULL) {
            printf("Memory allocation failed\n");
            return;
        }
        printf("Enter data for node %d: ", i);
        scanf("%d", &data);

        newNode->data = data;
        newNode->next = NULL;

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

        temp = newNode;
    }

    printf("\nLinked list created successfully\n");
}

void insertAtBeginning(int data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
```

D:\chethanDIP\sindlylinkedlist.c   C/C++   Windows (CR+LF)   WINDOWS-1252   Line 27, Col 6, Pos 445   Insert   Read/Write   default

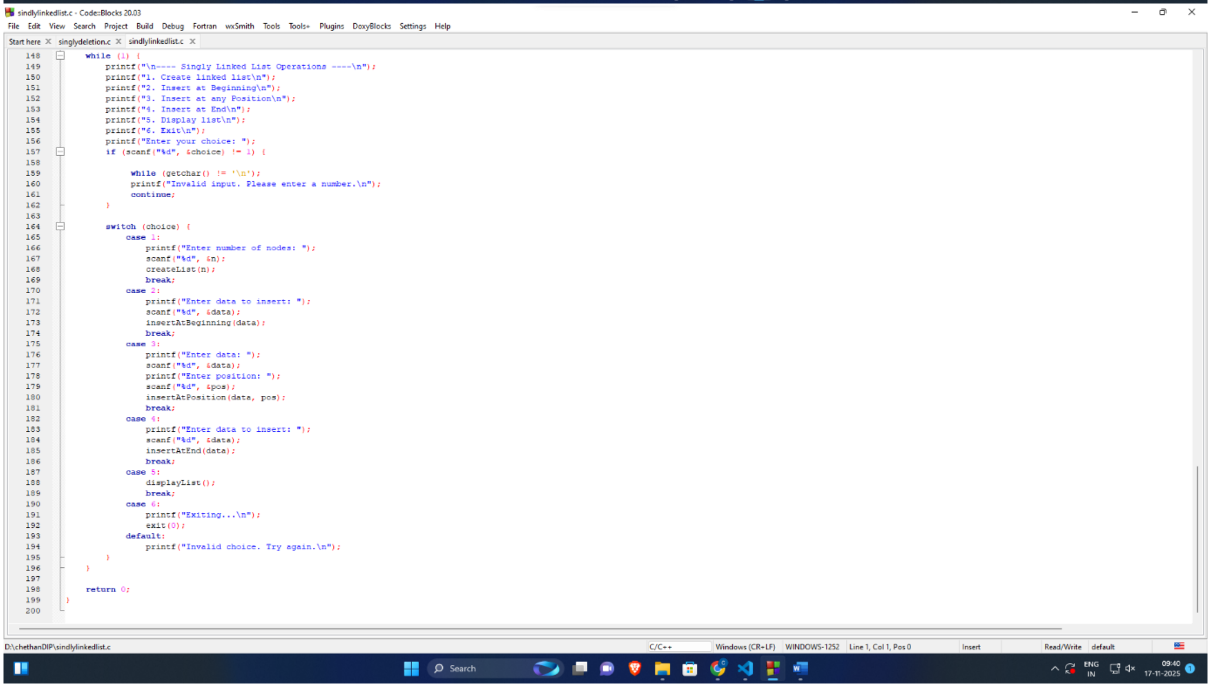

sindlylinkedlist.c - Code::Blocks 20.03

```c
void insertAtBeginning(int data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = head;
    head = newNode;
    printf("Node inserted at the beginning\n");
}

void insertAtEnd(int data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
    }
    else {
        struct node *temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }

    printf("Node inserted at the end\n");
}


void insertAtPosition(int data, int pos) {
    int i;
    struct node *newNode, *temp = head;

    if (pos < 1) {
        printf("Invalid position. Position must be 1 or greater.\n");
        return;
    }

    if (pos == 1) {
        insertAtBeginning(data);
        return;
    }


    for (i = 1; i < pos - 1 && temp != NULL; i++)
        temp = temp->next;

```

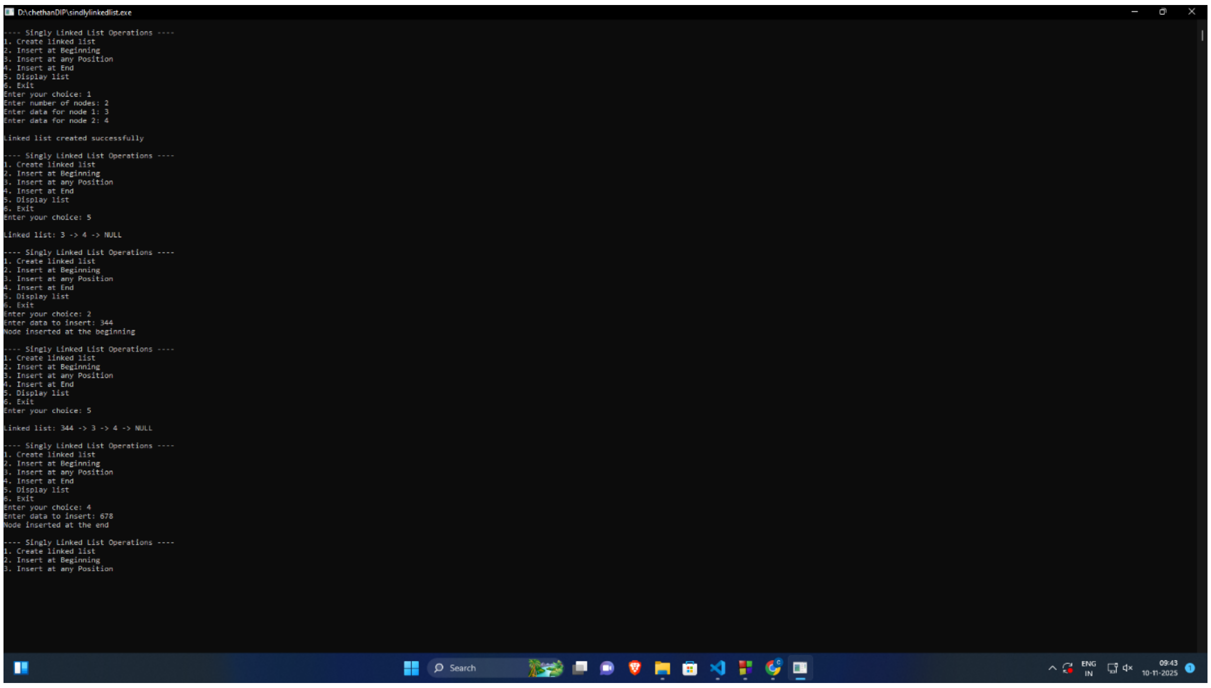
D:\chethanDIP\sindlylinkedlist.c   C/C++   Windows (CR+LF)   WINDOWS-1252   Line 27, Col 6, Pos 445   Insert   Read/Write   default


sindlylinkedlist.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
sindlylinkedlist.c
D:\chethanDIP\sindlylinkedlist.c
C/C++ Windows (CR+LF) WINDOWS-1252 Line 27, Col 6, Pos 445 Insert Read/Write default

```c
        temp = temp->next;


    if (temp == NULL) {
        printf("Position out of range: List is not long enough to reach position %d.\n", pos);
        return;
    }


    newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;


    newNode->next = temp->next;

    temp->next = newNode;

    printf("Node inserted at position %d\n", pos);
}

void displayList() {
    struct node *temp = head;

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

    printf("\nLinked list: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}



int main() {
    int choice, n, data, pos;

    while (1) {
        printf("\n---- Singly Linked List Operations ----\n");
        printf("1. Create linked list\n");
        printf("2. Insert at Beginning\n");
        printf("3. Insert at any Position\n");
        printf("4. Insert at End\n");
        printf("5. Display list\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
```


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

```c
    while (1) {
        printf("\n---- Singly Linked List Operations ----\n");
        printf("1. Create linked list\n");
        printf("2. Insert at Beginning\n");
        printf("3. Insert at any Position\n");
        printf("4. Insert at End\n");
        printf("5. Display list\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        if (scanf("%d", &choice) != 1) {

             while (getchar() != '\n');
             printf("Invalid input. Please enter a number.\n");
             continue;
        }

        switch (choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n);
                break;
            case 2:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtBeginning(data);
                break;
            case 3:
                printf("Enter data: ");
                scanf("%d", &data);
                printf("Enter position: ");
                scanf("%d", &pos);
                insertAtPosition(data, pos);
                break;
            case 4:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtEnd(data);
                break;
            case 5:
                displayList();
                break;
            case 6:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Try again.\n");
        }
    }

    return 0;
}
```



D:\chethanDIP\sindlylinkedlist.exe

```
---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 1
Enter number of nodes: 2
Enter data for node 1: 3
Enter data for node 2: 4

Linked list created successfully

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 5

Linked list: 3 -> 4 -> NULL

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 2
Enter data to insert: 344
Node inserted at the beginning

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 5

Linked list: 344 -> 3 -> 4 -> NULL

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 4
Enter data to insert: 678
Node inserted at the end

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
```

DATE: 10/11/25 PAGE:

11) write a program to impliment ~~[illegible]~~ singly linked list with the following a) create a linked list b) Insertion of a node at first position & any position & end of the list c) Display the contents of linked list

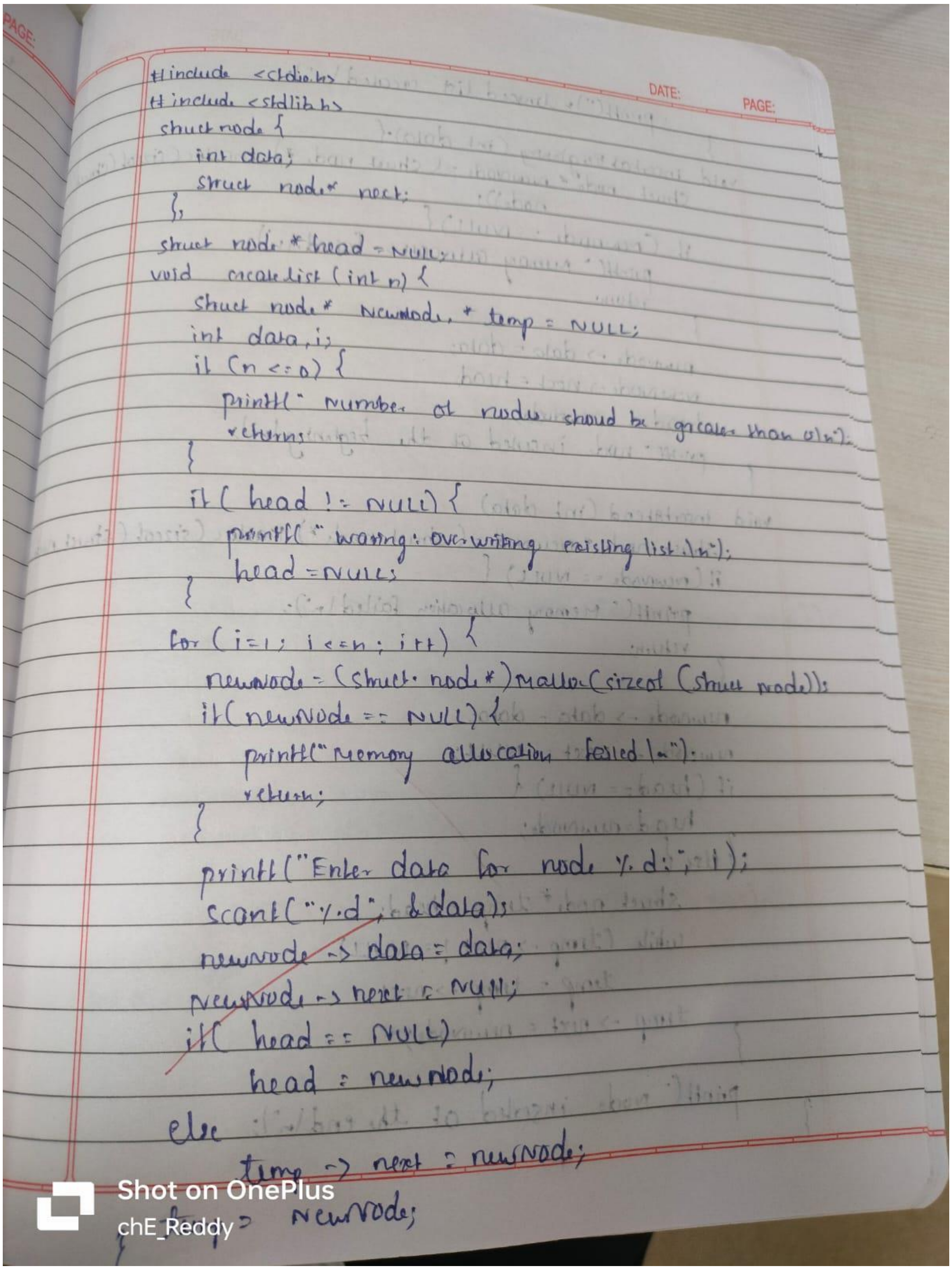
pseudocode

```
Structure node
    data
    next -> node
end structure
=> empty linked list = head <- null
=> New node create
        create Node (value)
        Newnode <- allocate memory for node
        newnode.data <- value
        newnode.next <- null
        return node
    end function
=> Insert at first position
    firstposition (value)
        newnode <- create node (value)
        newnode.next <- head
        head <- newnode
    end function
=> Insert at end
    insertatend (value)
        newnode -> createnode (value)
        if head = null then
            head <- newnode
            return
        endif
        temp <- head
```

DATE:

```
        while temp.next ≠ null
            temp ← temp.next
        end while
        temp.next ← newnode
    end function
=> Insert at any position:
    Insertatanyposition ( targetvalue, newvalue)
        temp ← head
        while temp ≠ null and temp.data ≠ targetvalue
                temp ← temp.next
        end while
        If temp = null then
            print "target not found"
            return
        endif
        newnode ← createnode (newvalue)
        newnode.next ← temp.next
        temp.next ← newnode
    end
=> Display
    Display ()
    temp ← head
    while temp ≠ null
        print temp.data
        temp ← temp.next
    end while
end function
```

[illegible]



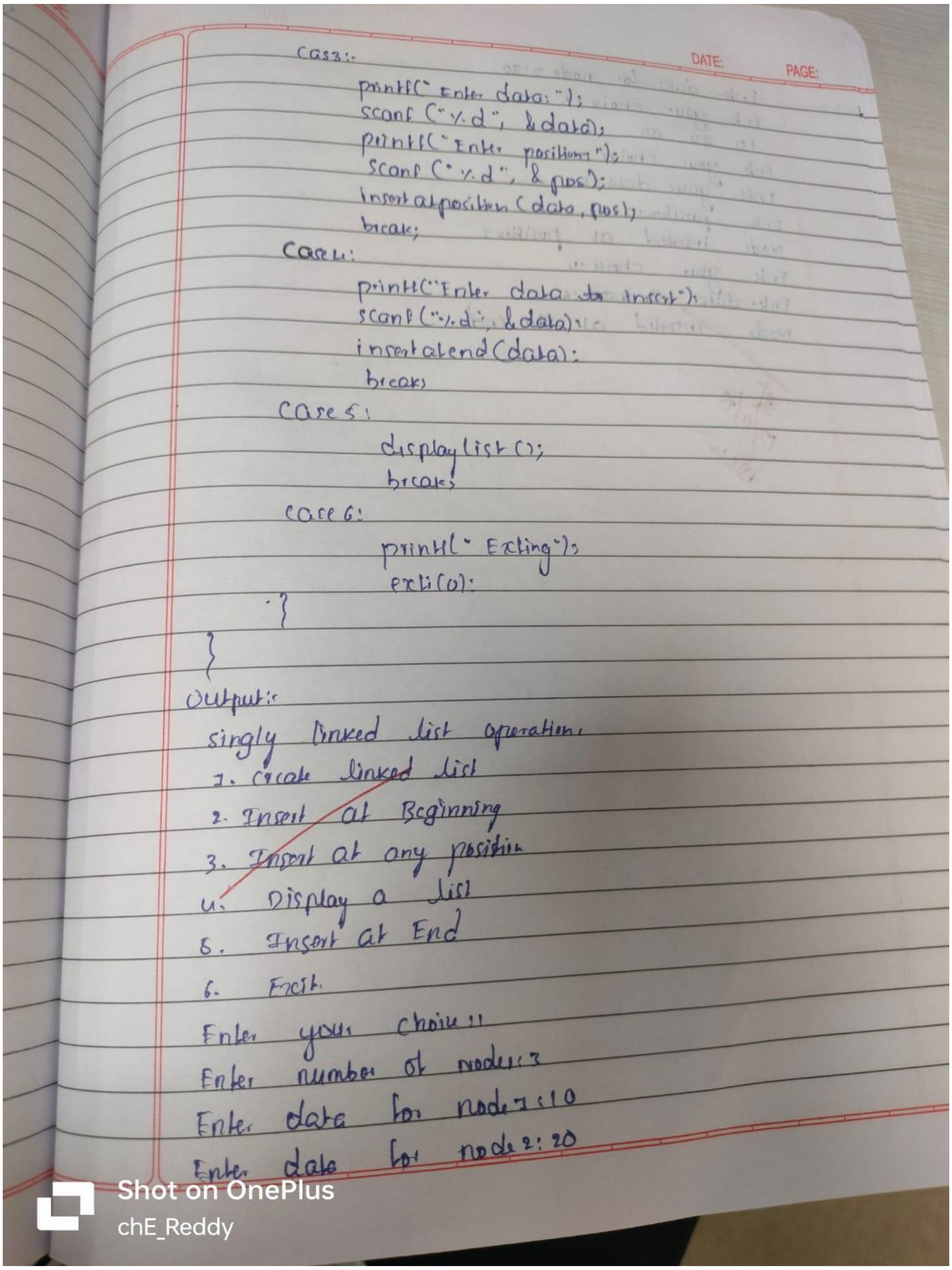
```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node* next;
};
struct node* head = NULL;
void createlist (int n) {
    struct node* Newnode, * temp = NULL;
    int data,i;
    if (n <= 0) {
        printf(" number of nodes shoud be greater than 0\n");
        return;
    }
    if( head != NULL) {
        printf(" warning: overwriting existing list.\n");
        head = NULL;
    }
  for (i=1; i<=n; i++) {
     newnode = (struct node*)malloc(sizeof (struct node));
     if(newNode == NULL) {
         printf("memory allocation failed \n");
         return;
     }
     printf("Enter data for node %d:", i);
     scanf("%d", &data);
     newnode -> data = data;
    NewNode -> next = NULL;
    if( head == NULL)
         head = newNode;
  else
         temp -> next = newNode;
     temp = NewNode;
}
```

```
    printf("\n linked list created\n");
}
void insertatBeginning (int data){
    struct node* newnode = (struct node *) malloc (sizeof(
                    node));
    if (newnode ==NULL){
        printf(" Memory allocation failed.\n");
        return;
    }
    newnode -> data = data;
    newNode -> next = head;
    head = newnode;
    printf(" node inserted at the beginning\n");
}
void insertAtend (int data)
    struct node * newnode = (struct node *) malloc (sizeof (struct
    if (newnode == NULL) {
        printf(" Memory allocation failed\n");
        return;
    }
    newnode -> data = data;
    newnode -> next = NULL;
    if (head == NULL) {
        head = newnode;
    } else {
        struct node* temp = head;
        while (temp -> next != NULL)
            temp = temp -> next;
        temp -> next = newnode;
    }
    printf(" node inserted at the end\n");
```

```
void insertatposition(int data, int pos) {
     int i;
     struct node * newnode, * temp = head;
     if (pos < 1) {
         printf("Invalid position\n");
         return;
     }
     if (pos == 1) {
         insertAt Beginning (data);
         return;
     }
     for (i=1; i<pos-1 && temp != NULL; i++)
         temp = temp->next;
     if temp == NULL) {
         printf("position out of range(\n", pos);
         return;
     }
     newNode = (struct node*) Malloc (sizeof (struct node));
     if (newNode == NULL){
         printf (" Memory allocation failed \n");
         return;
     }
     ~~newNode -> data = data;~~
     ~~newNode -> next = temp->next;~~
     temp -> next = newNode;
     printf(" Node inserted at position\n", pos);
}
void  displaylist() {
     struct node * temp = head;
     if (head == NULL) {
         printf(" list is empty\n");
         return;
     }
```

```
        printf("\nLinked list:");
        while (temp != NULL){
            printf("%d ->", temp -> data);
            temp = temp -> next;
        }
        printf("NULL\n");
    }
int main () {
    int choice, n, data, pos;
    while (1) {
        printf("\n1. create linked list\n");
        printf("2. Insert at Beginning\n");
        printf("3. Insert at any position\n");
        printf("4. Insert at End\n");
        printf("5. Display list\n");
        printf("6. Exit\n");
        if (scanf("%d", &choice) != 1) {
            while (getchar () != '\n');
            printf("Invalid Input. please enter a number\n");
            continue;
        }
        switch (choice) {
            case1:
                printf("Enter number of nodes:");
                scanf("%d", &n);
                createList (n);
                break;
            case2:
                printf("Enter data to insert");
                scanf("%d", &data);
                insertatBeginning (data);
                break;
```

Shot on OnePlus
chE_Reddy



DATE: PAGE:

```
    case3:-
        printf("Enter data:");
        scanf("%d", &data);
        printf("Enter position:");
        scanf("%d", &pos);
        insertatposition(data, pos);
        break;
    case4:
        printf("Enter data to insert");
        scanf("%d", &data);
        insertatend(data);
        break;
    case5:
         displaylist();
         break;
    case6:
         printf(" Exiting");
         exit(0);
    }
}
```

Output:-

```
singly linked list operation
1. Create linked list
2. Insert at Beginning
3. Insert at any position
4. Display a list
5. Insert at End
6. Exit.
Enter your choice:1
Enter number of nodes:3
Enter data for node1:10
Enter data for node2:20
Enter data
Enter data for node 3:30
Enter your choice:5
 10  20  30
Enter your choice:3
Enter your data:4
Enter position:3
Node inserted at position 3
Enter your choice:4
Enter data to insert:6
Node inserted at the end
```

12/4/25 [illegible]