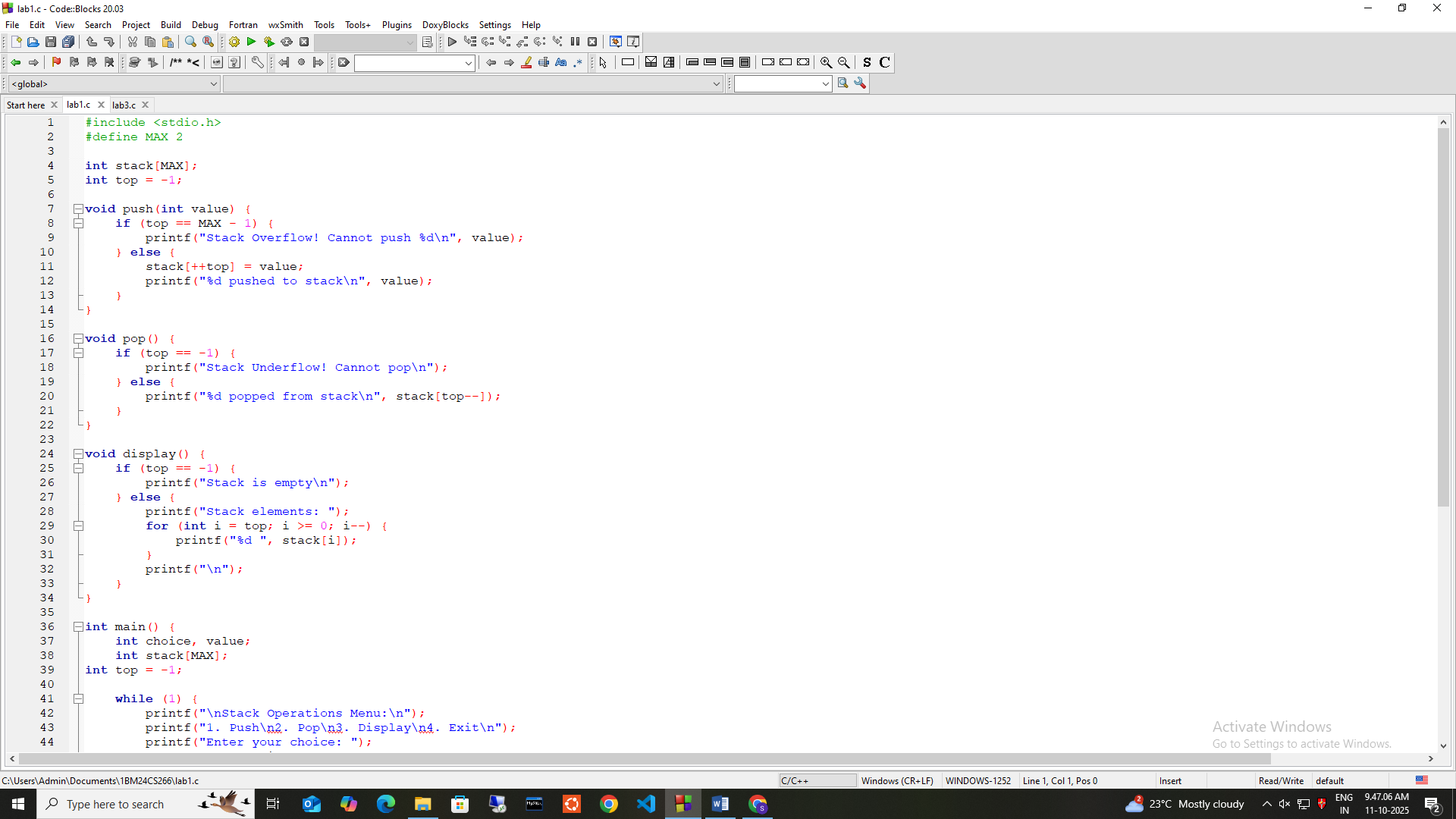
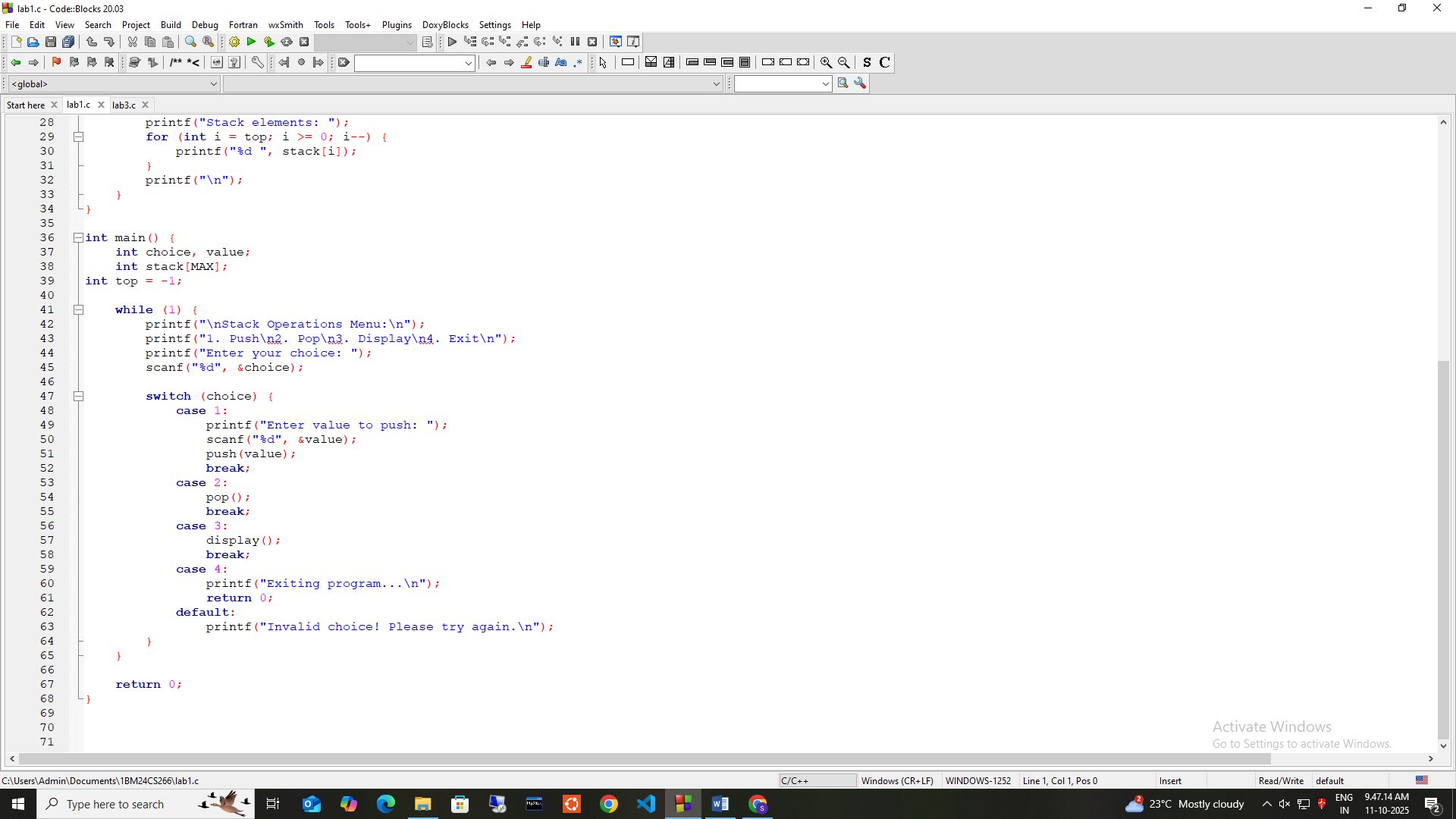
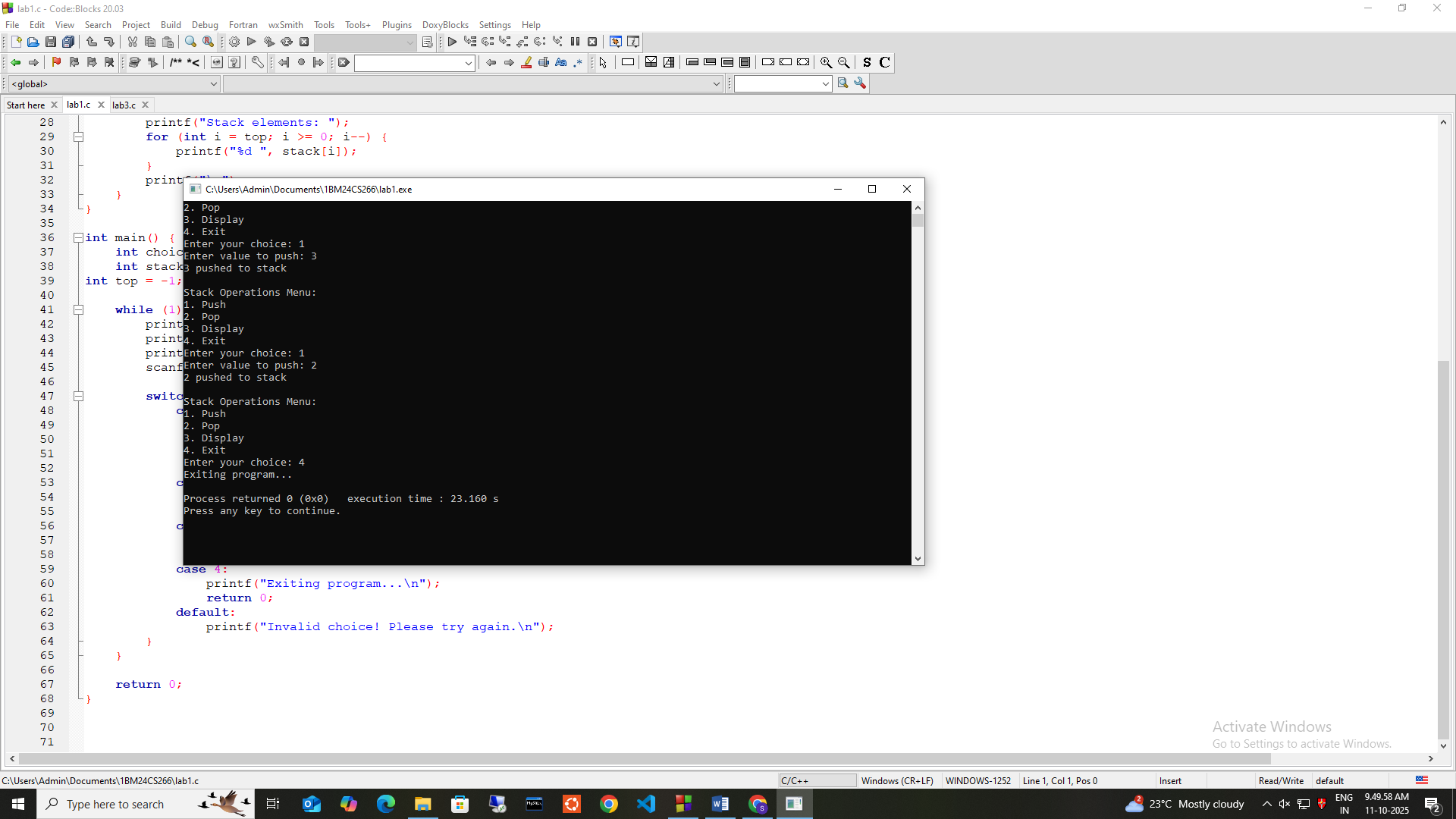
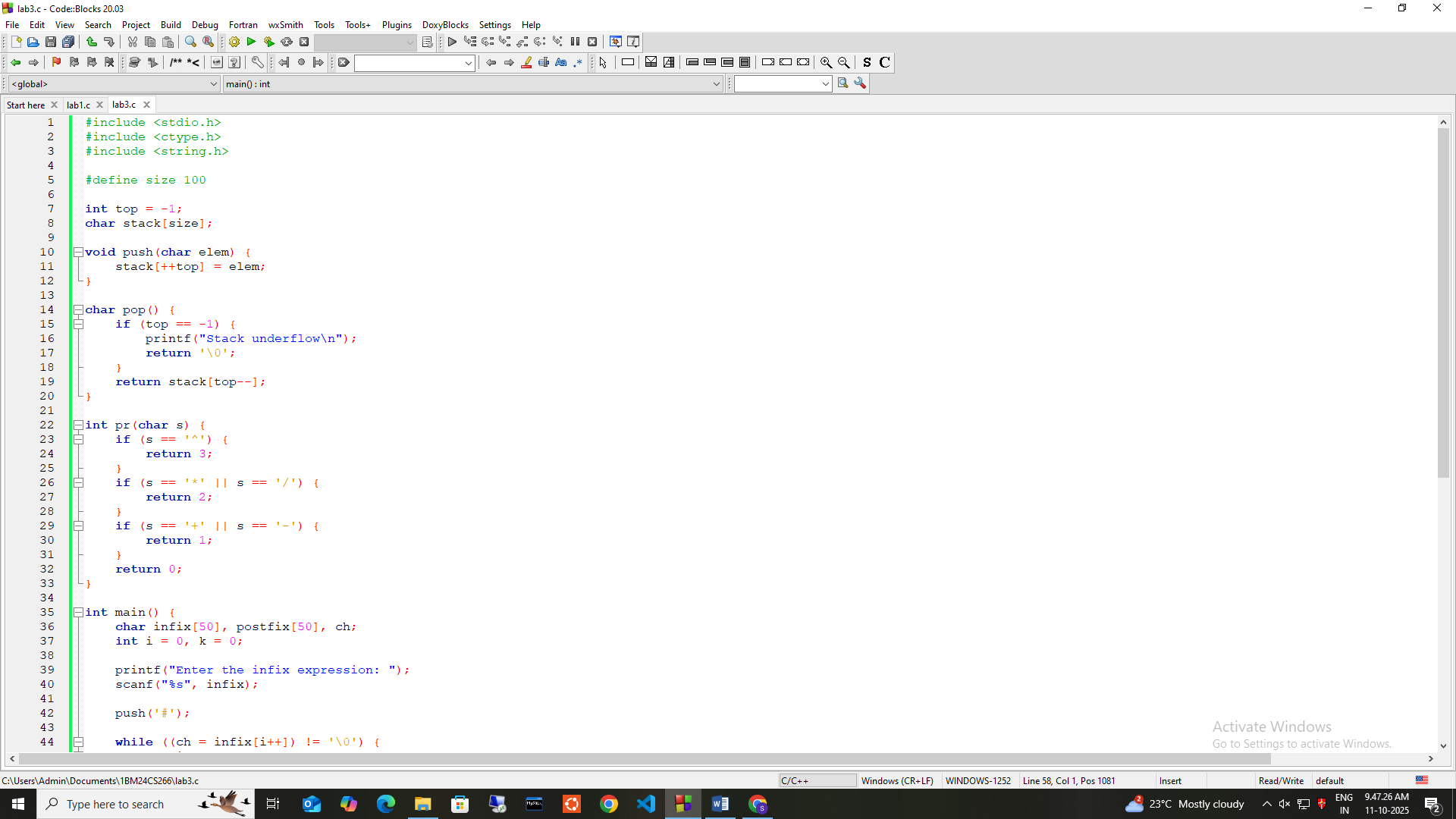
**1] Stimulate push pop display of stack**

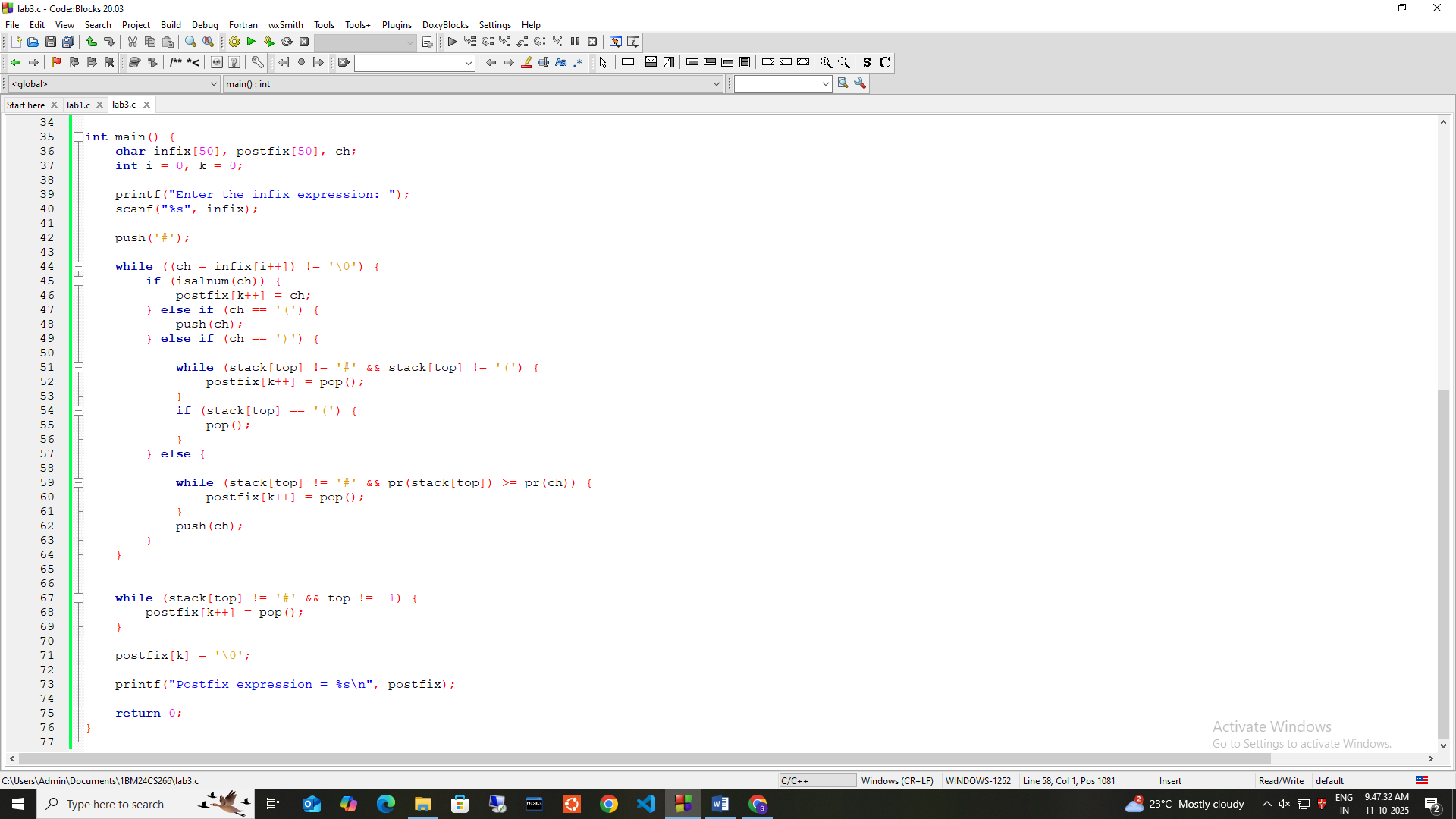
****

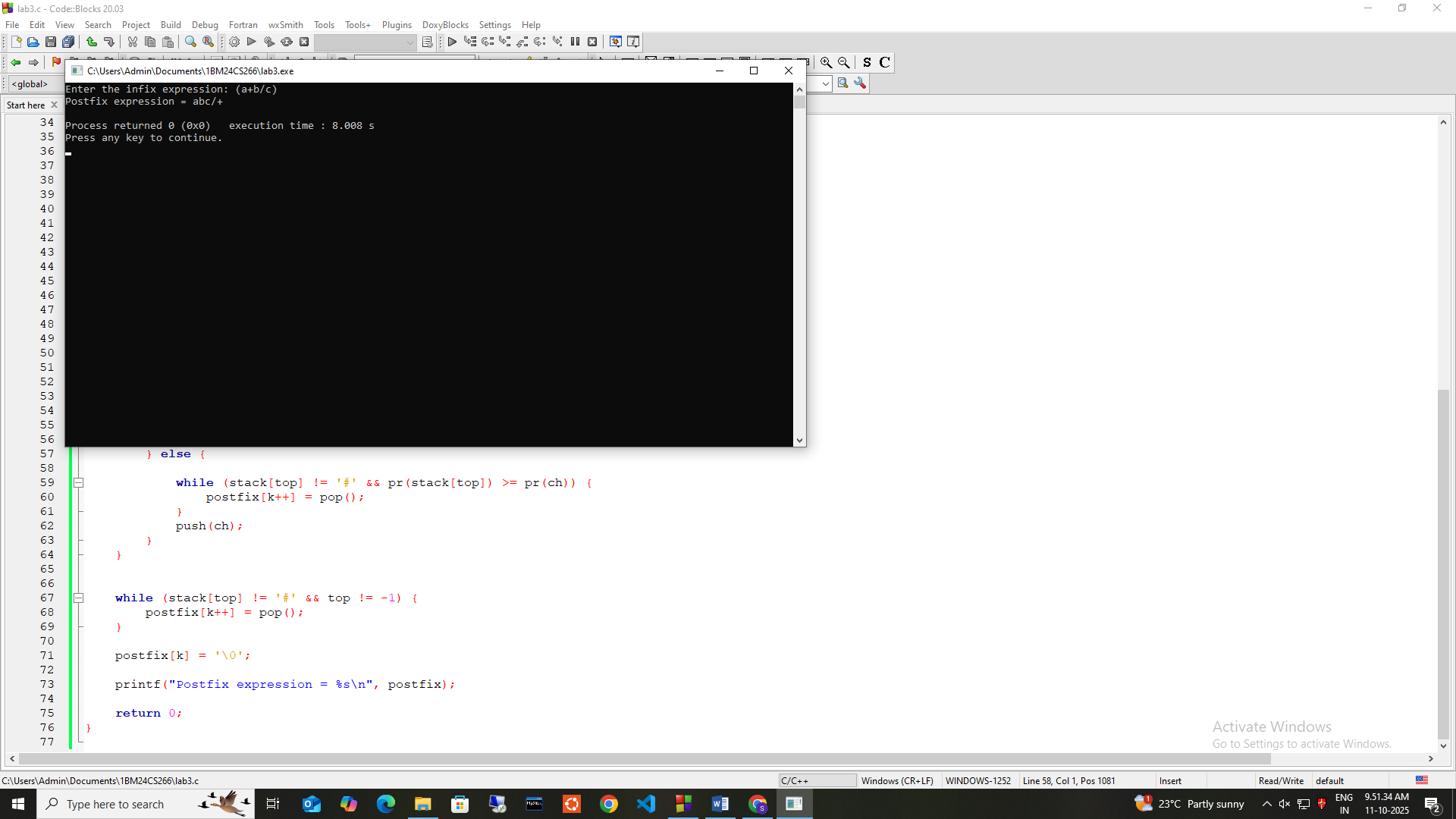
****

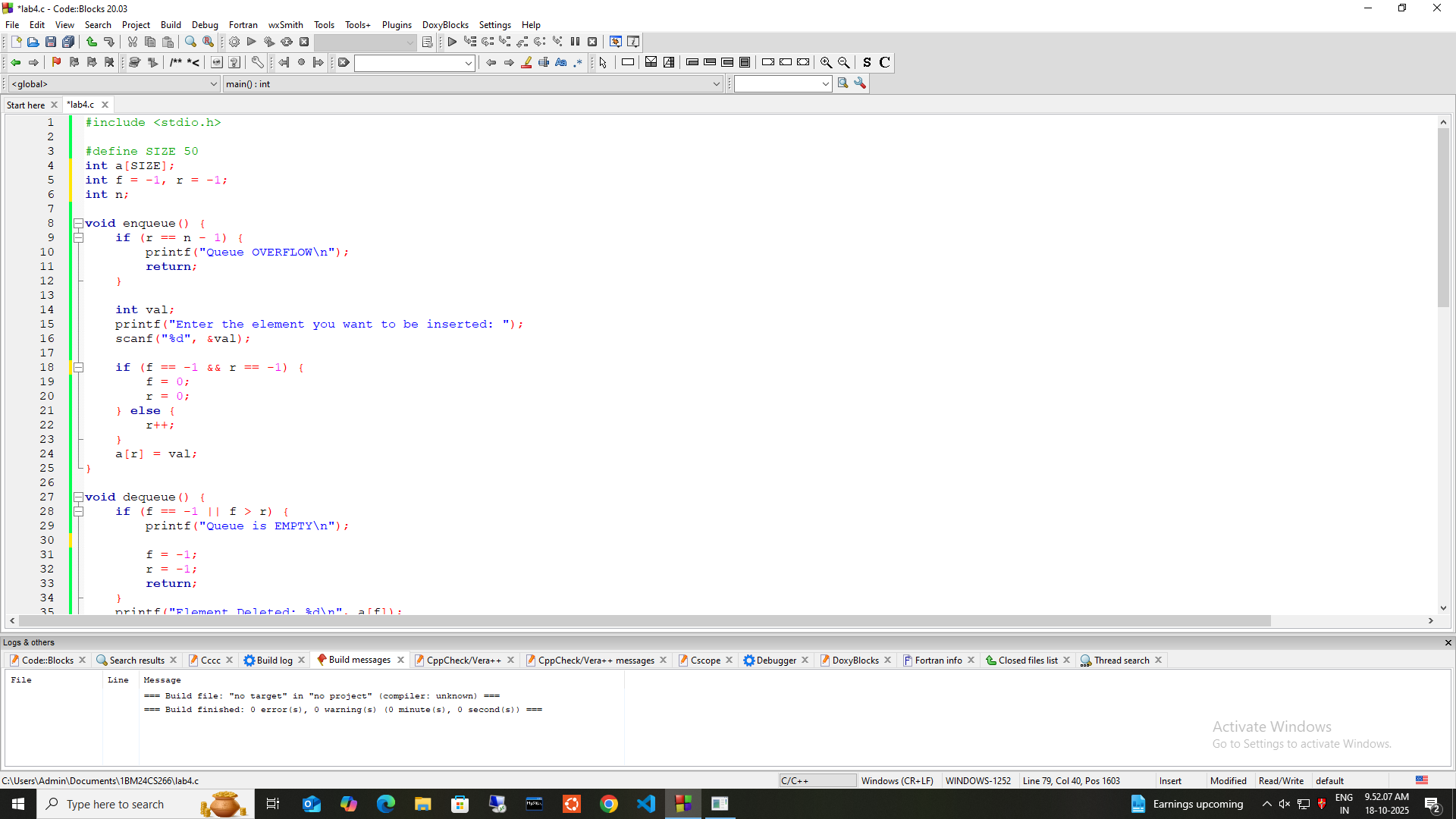
****

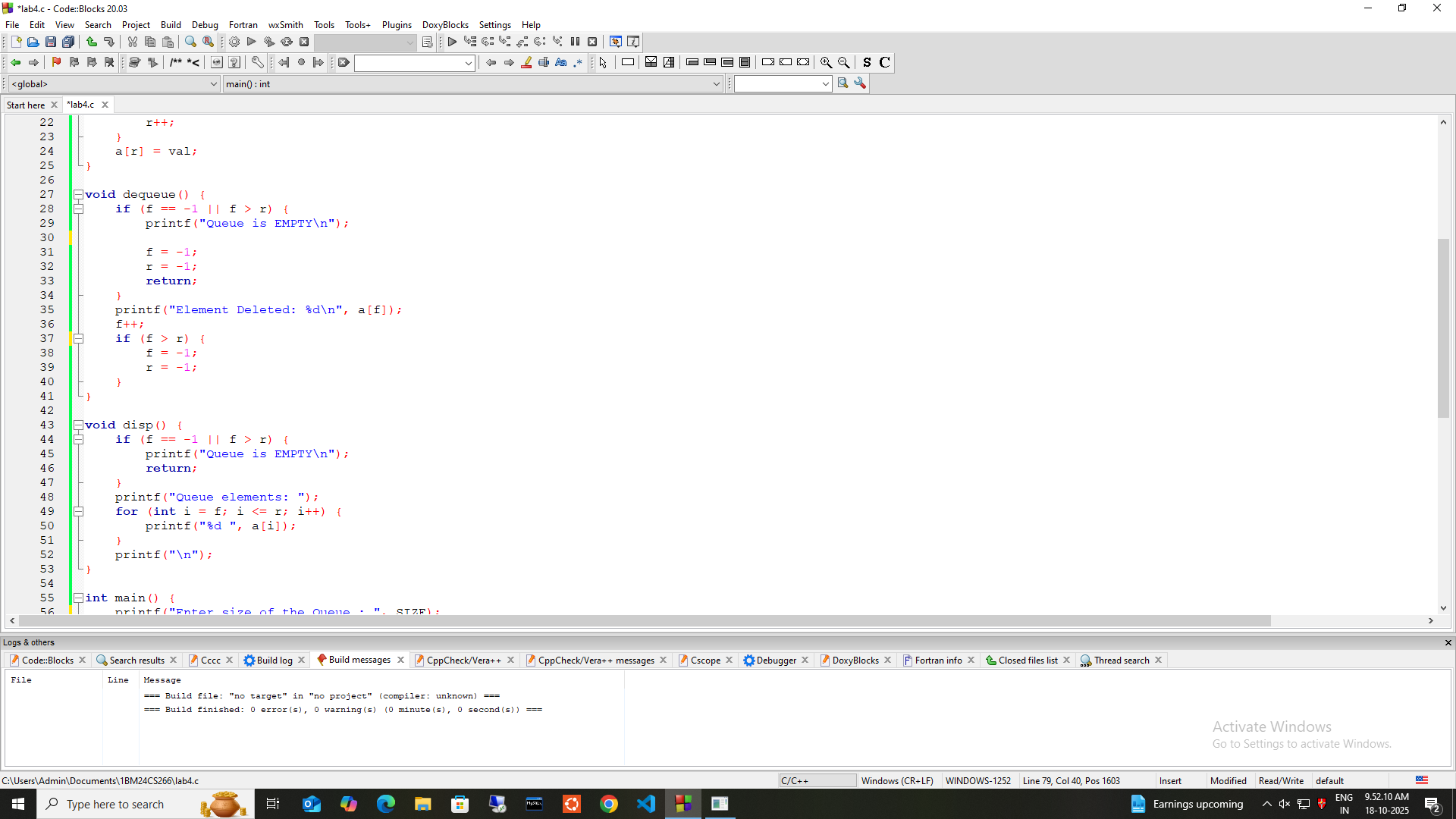
**2]infix to postfix**

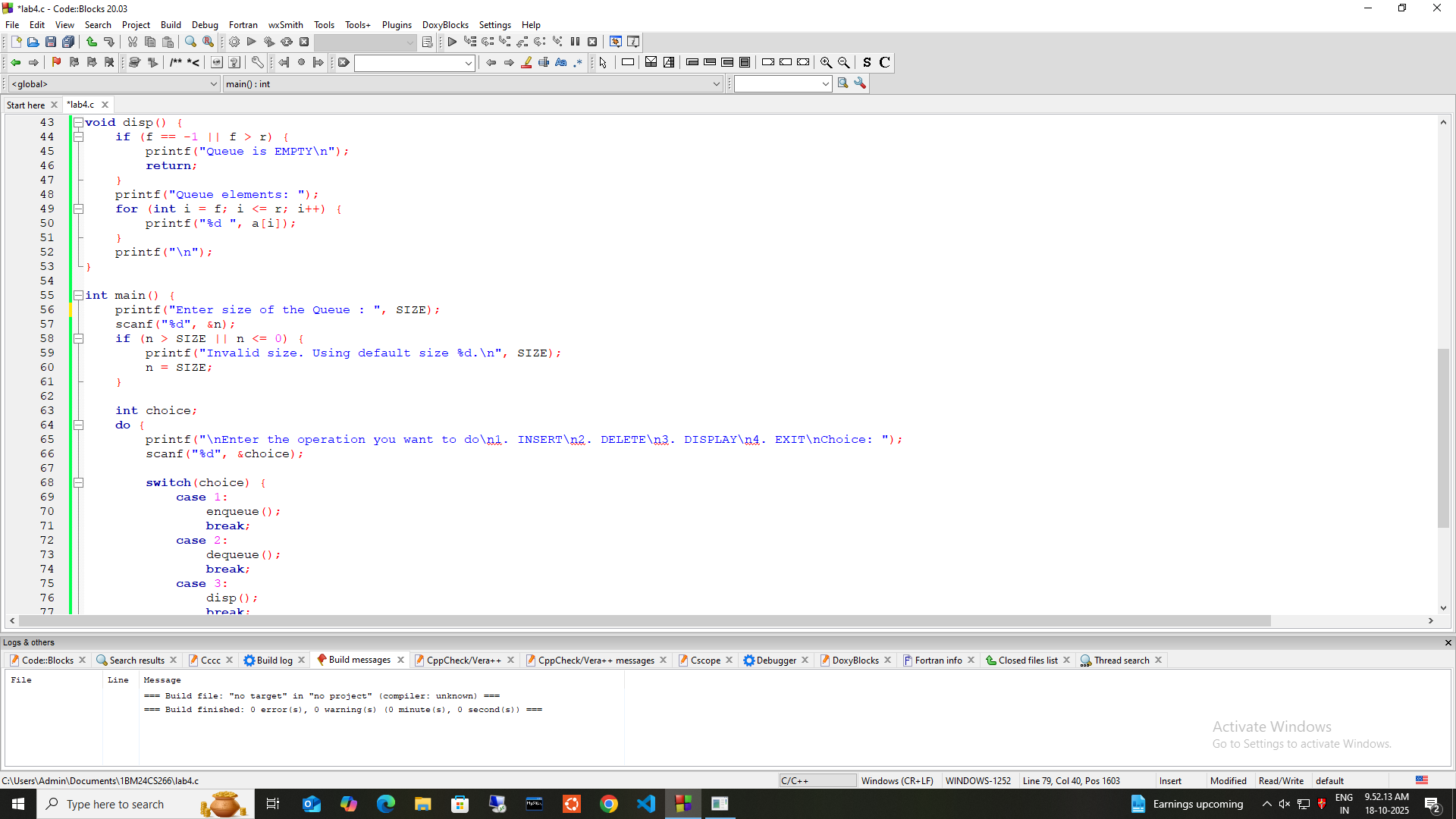
****

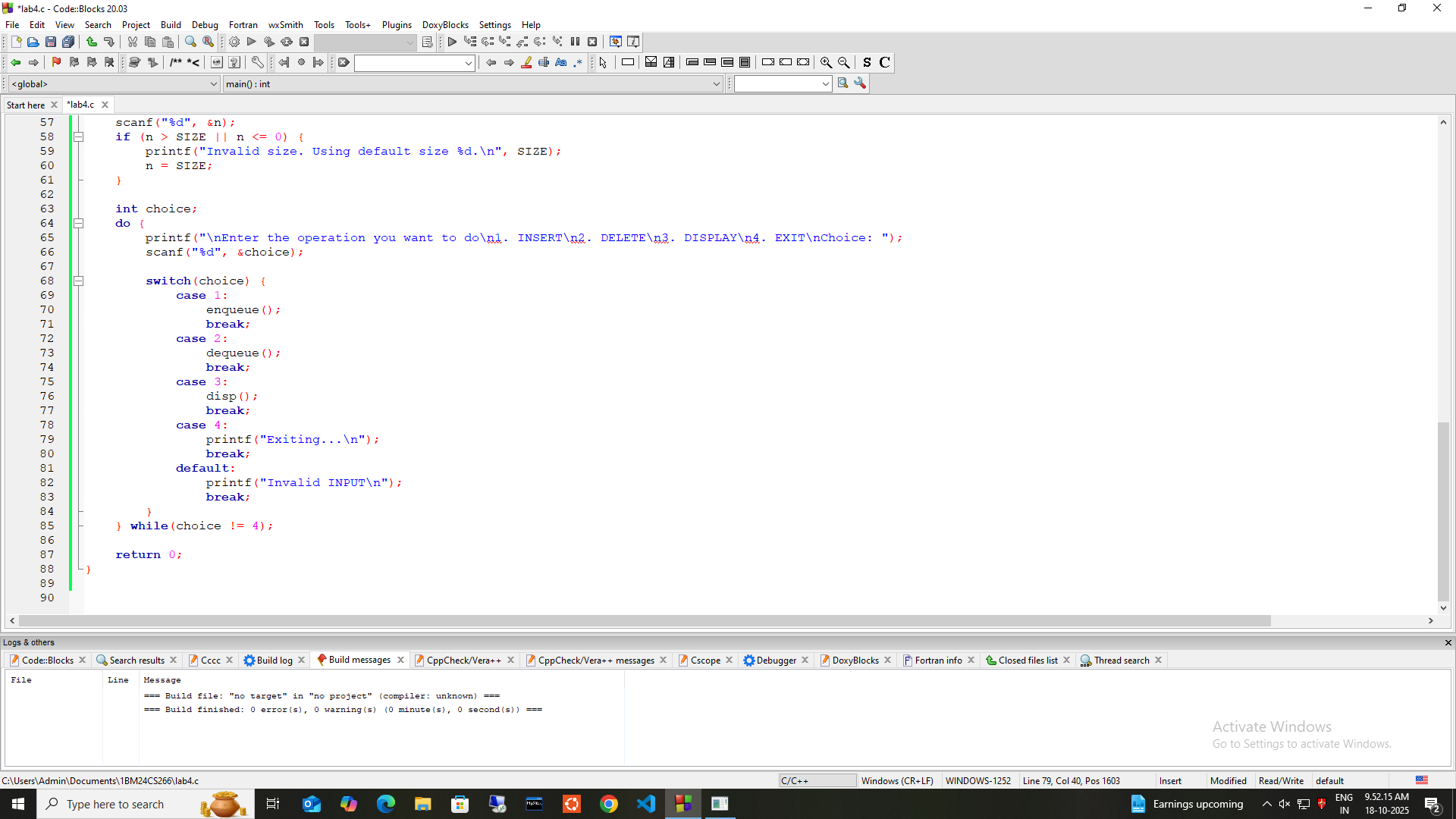
****

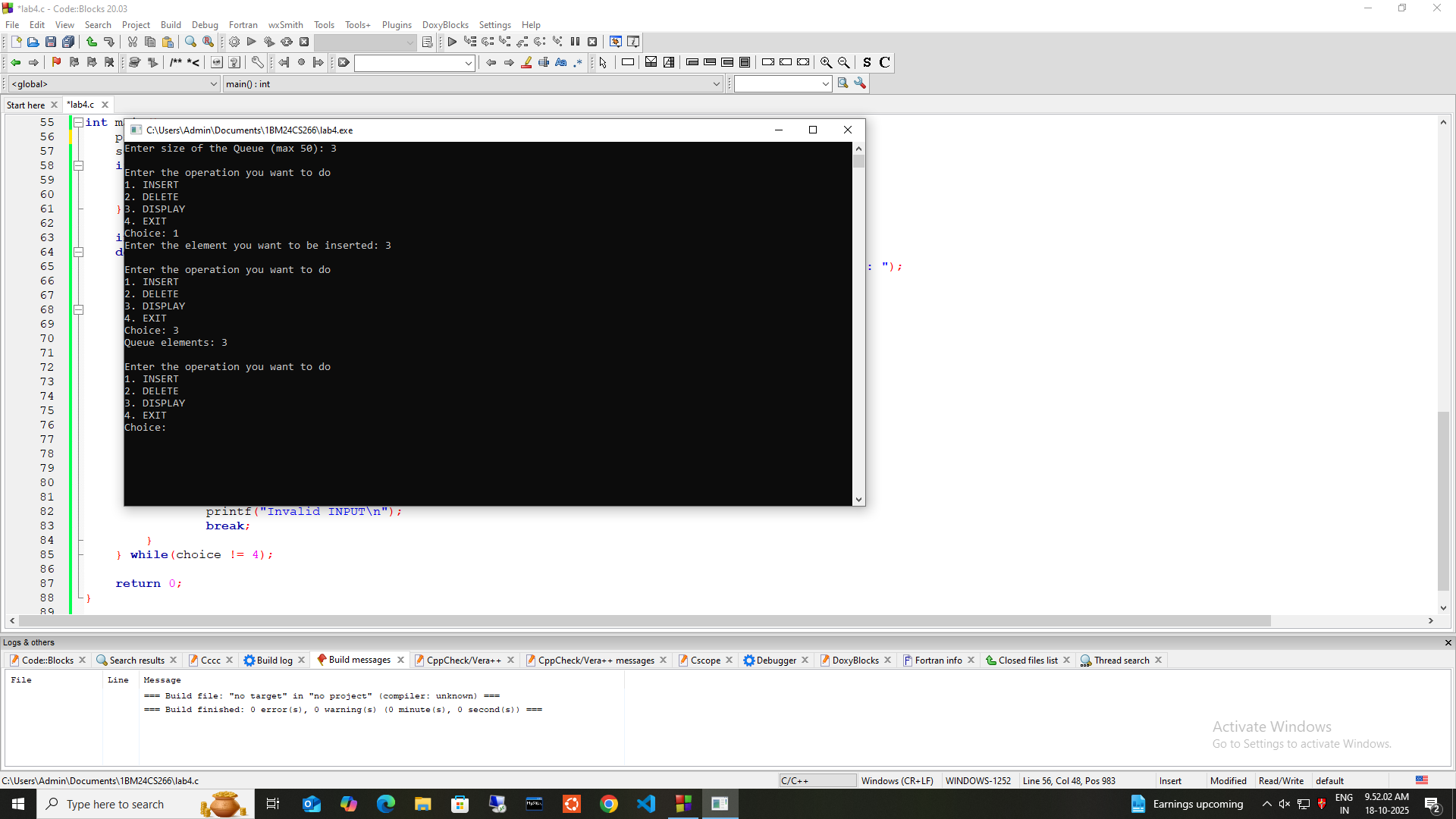
****

**3.a] Insert,Delete,Display on Que**

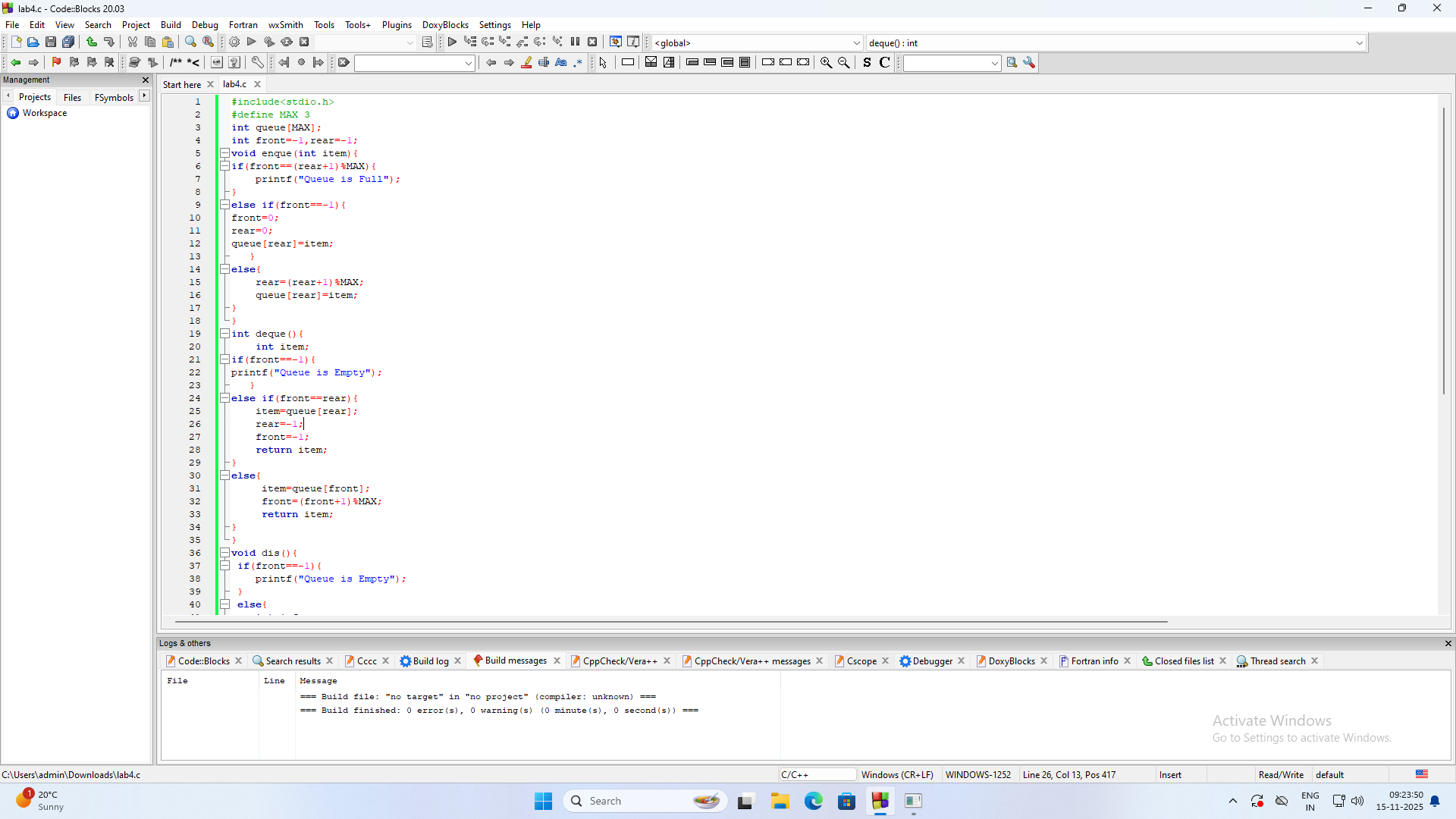
****

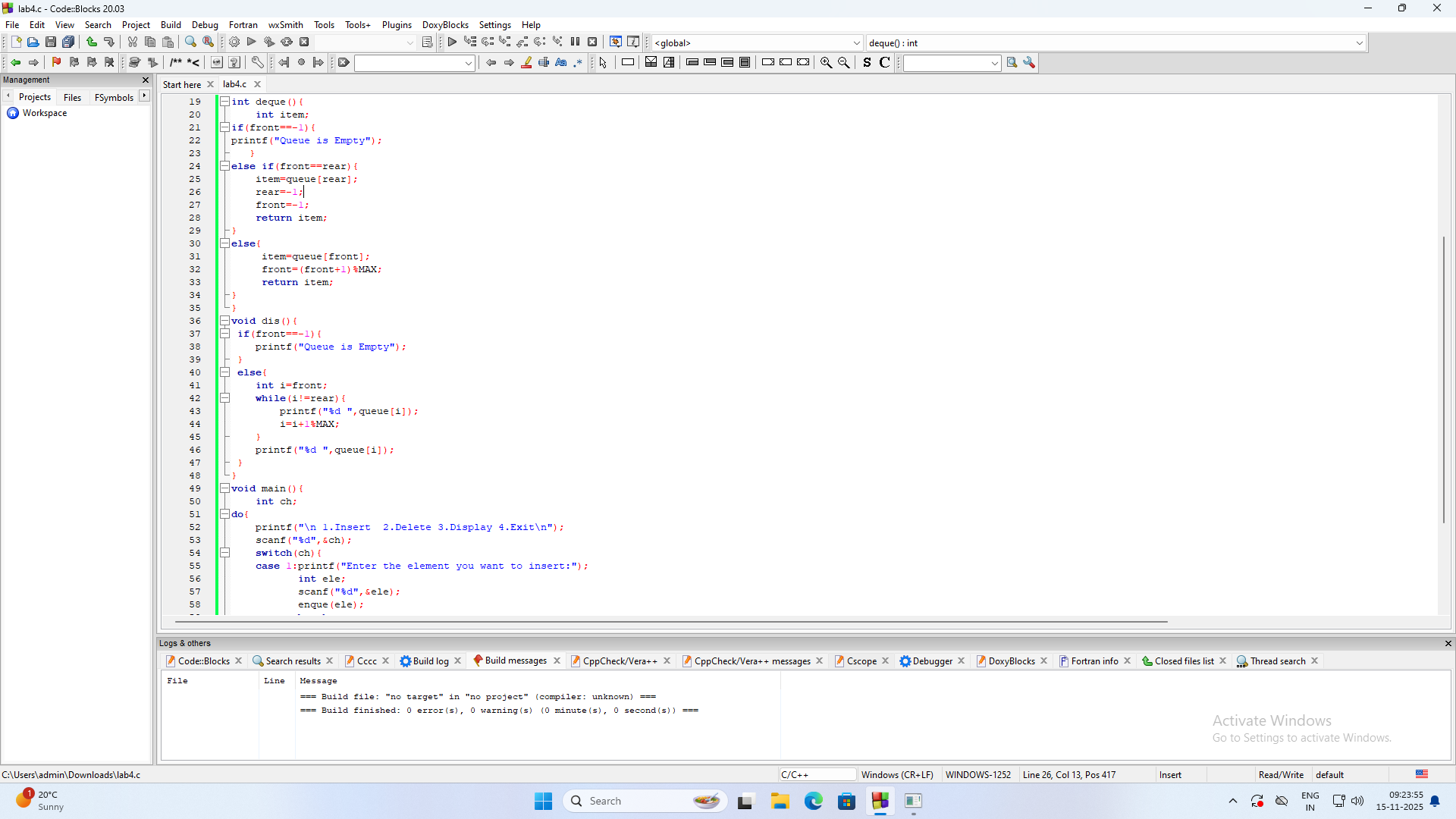
****

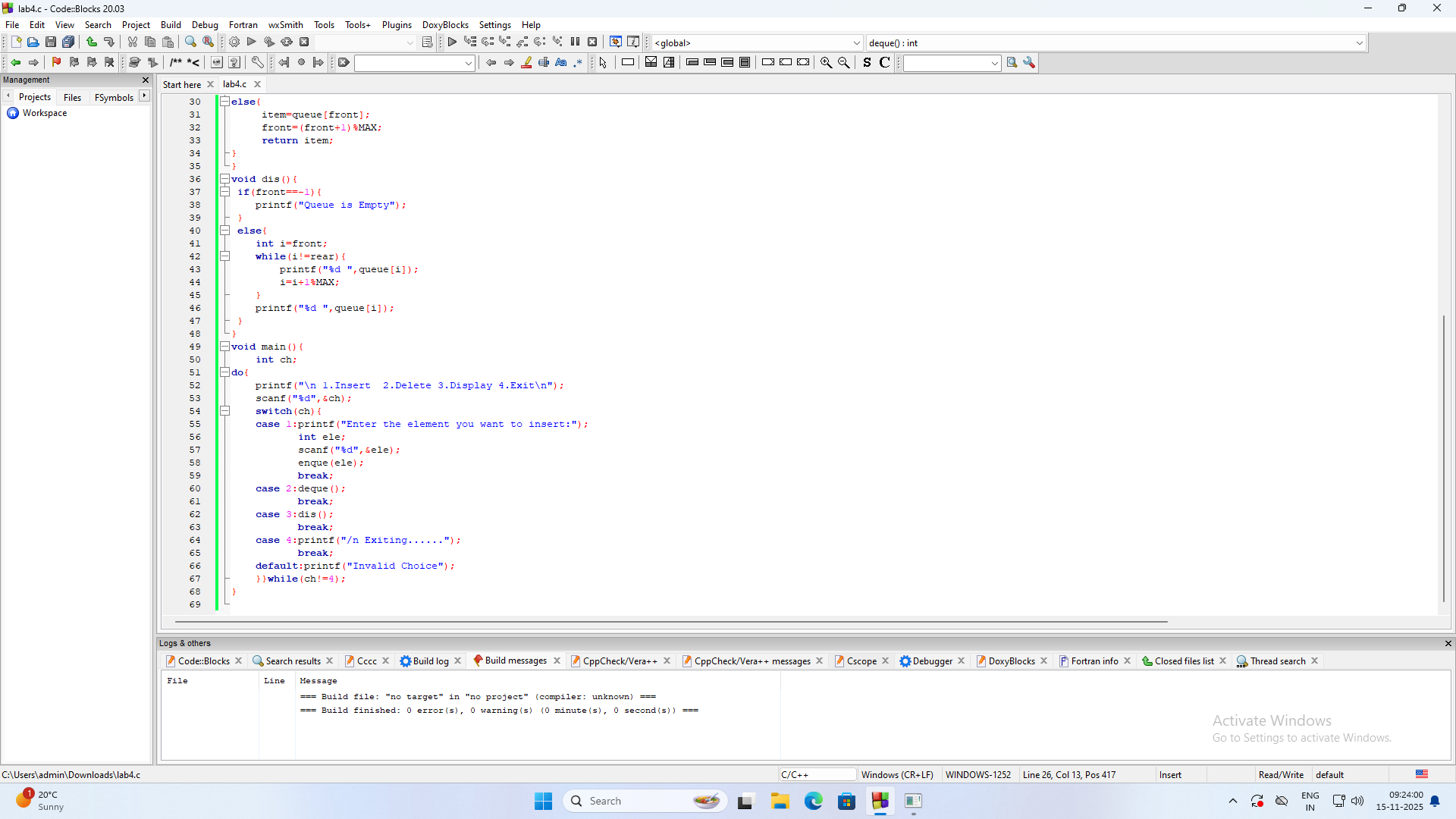
****

****

**3.b]Circular Que**

****

****

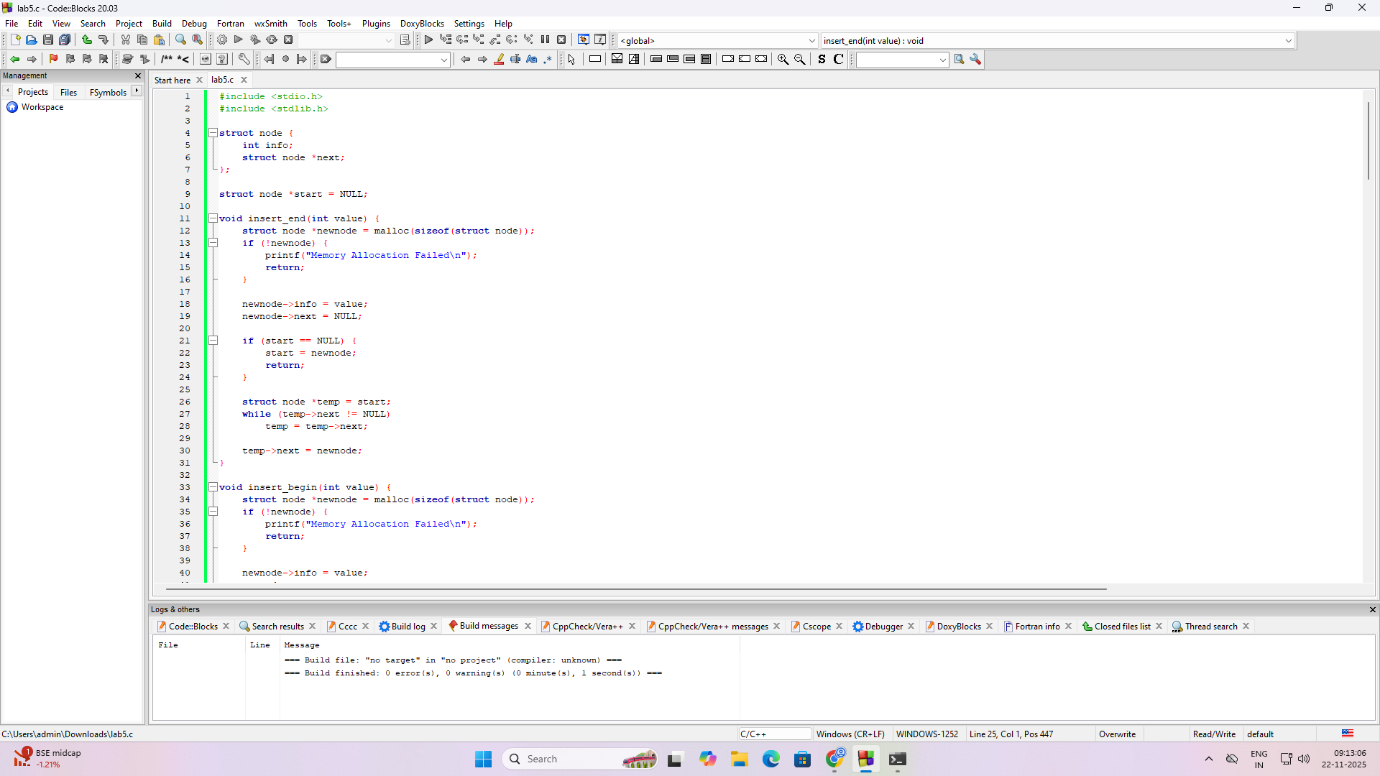
**4&5] WAP to Implement Singly Linked List with following operations**

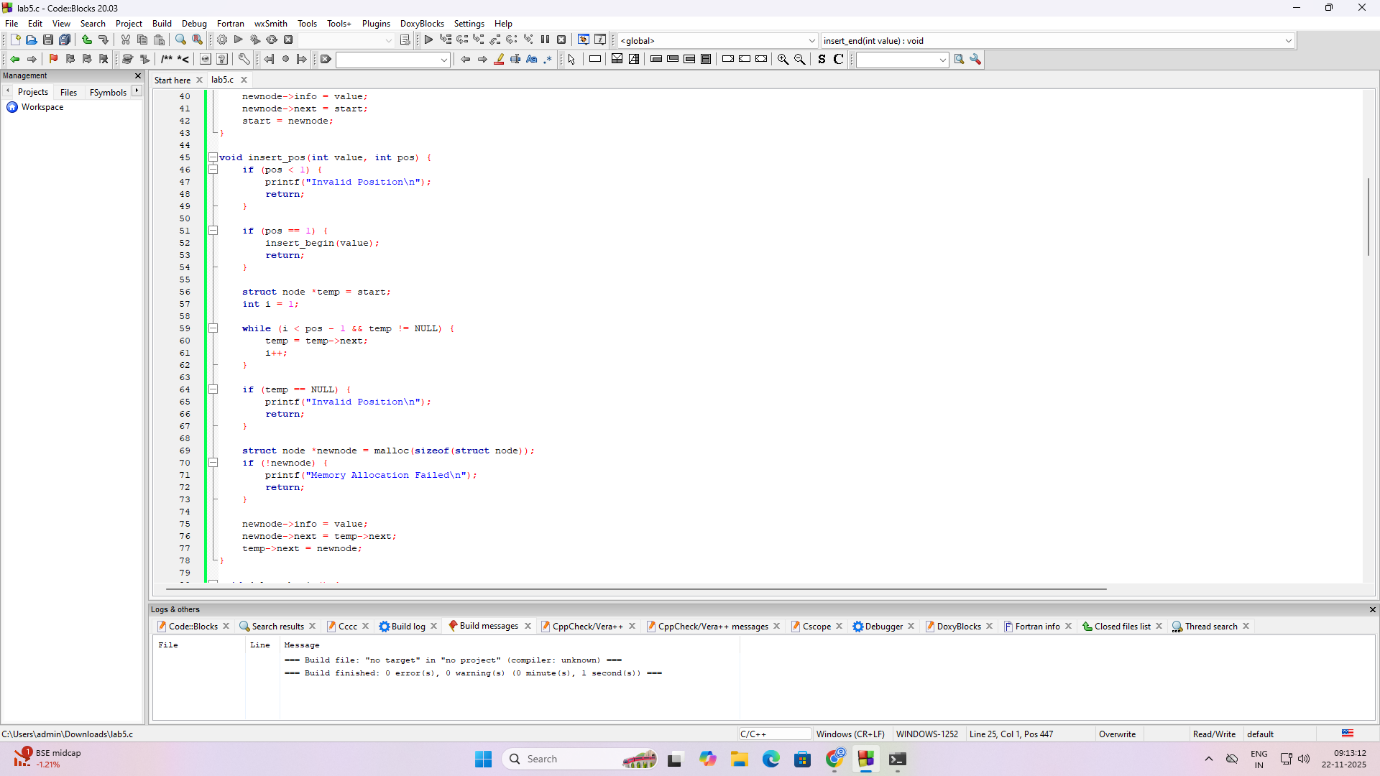
**a) Create a linkedlist.**

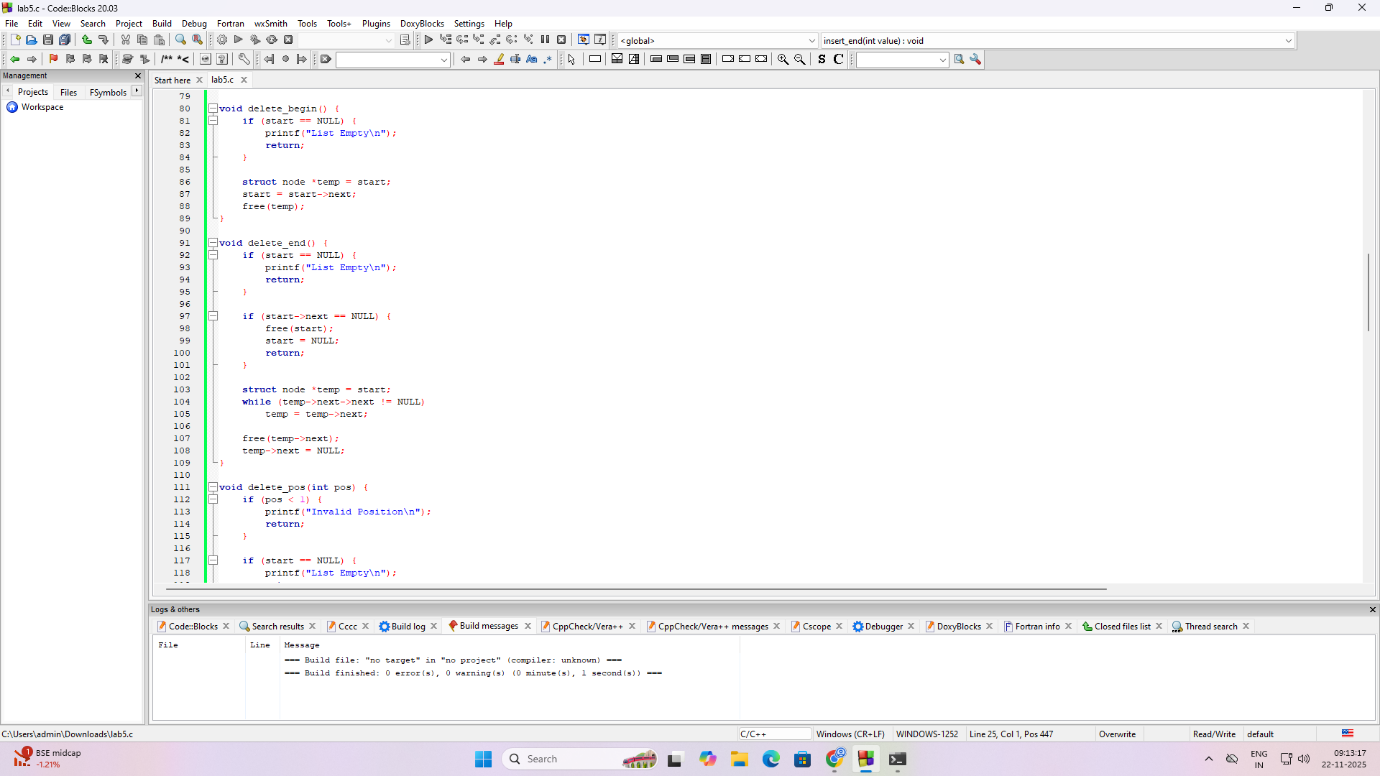
**b) Insertion of a node at first position, at any position and at end of list.**

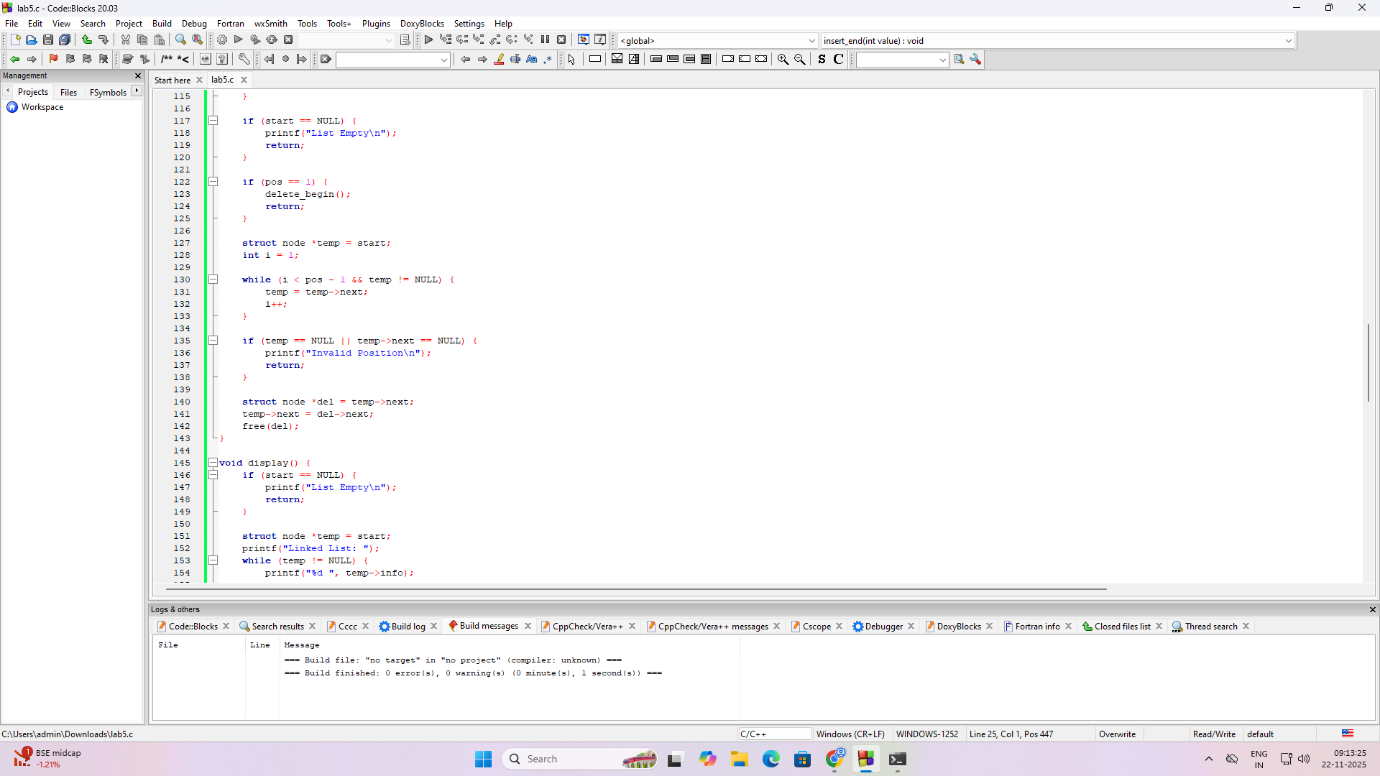
c**) Deletion of first element, specified element and last element in the list.**

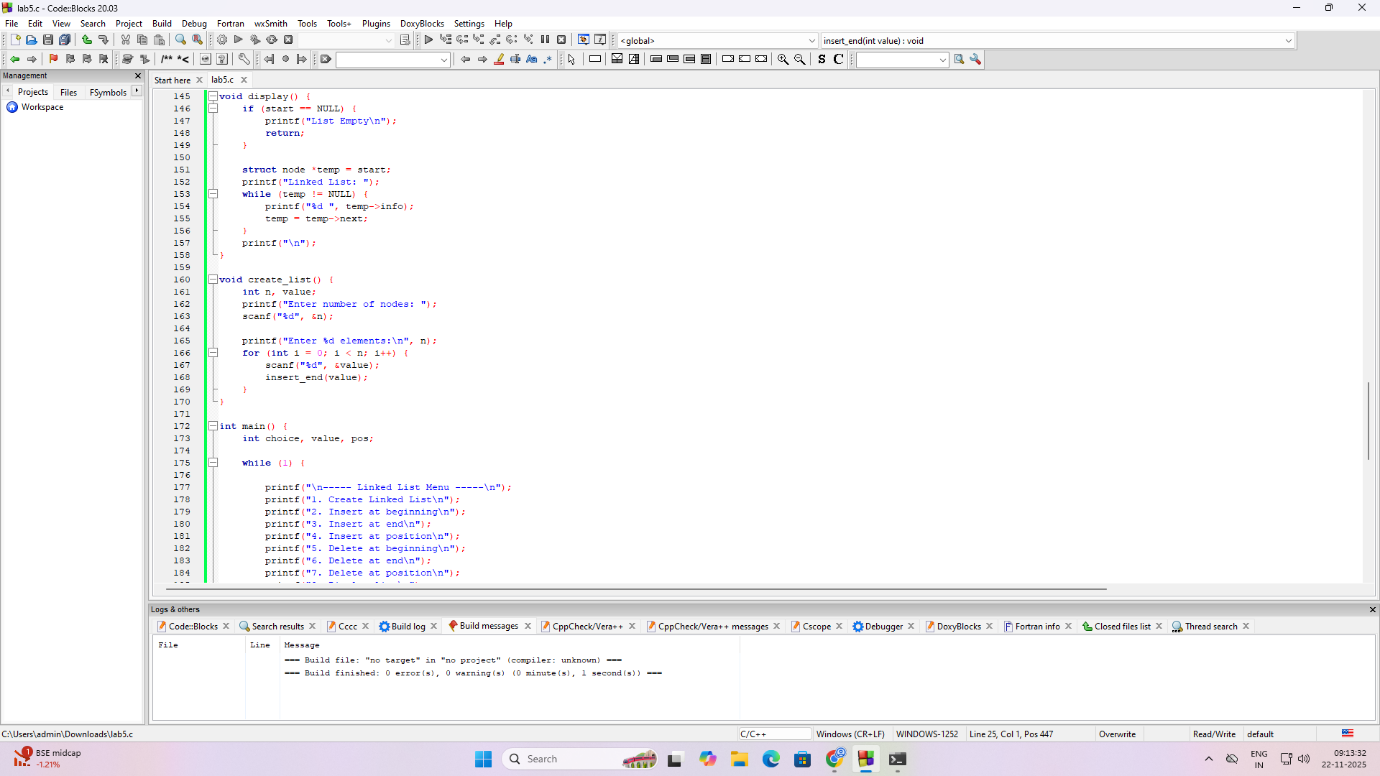
**d) Display the contents of the linked list.**

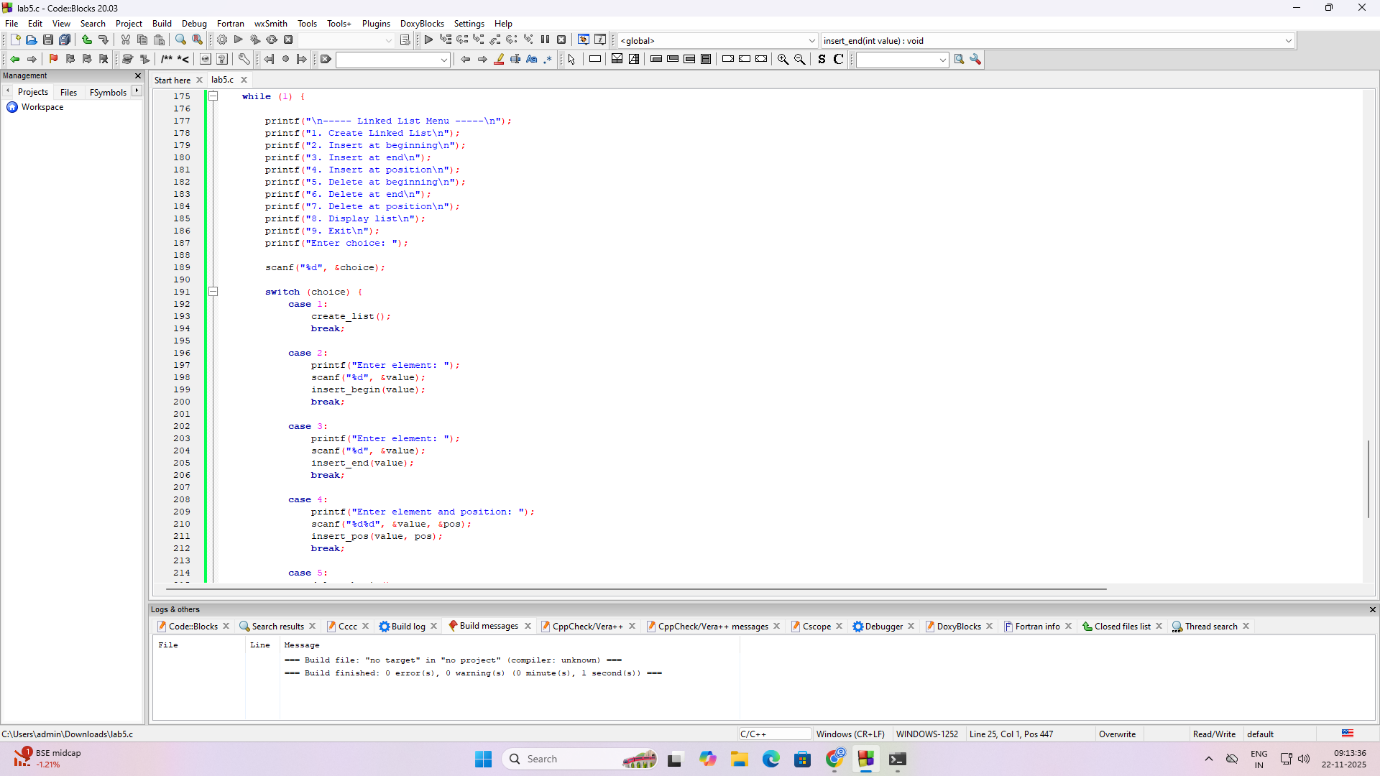
****

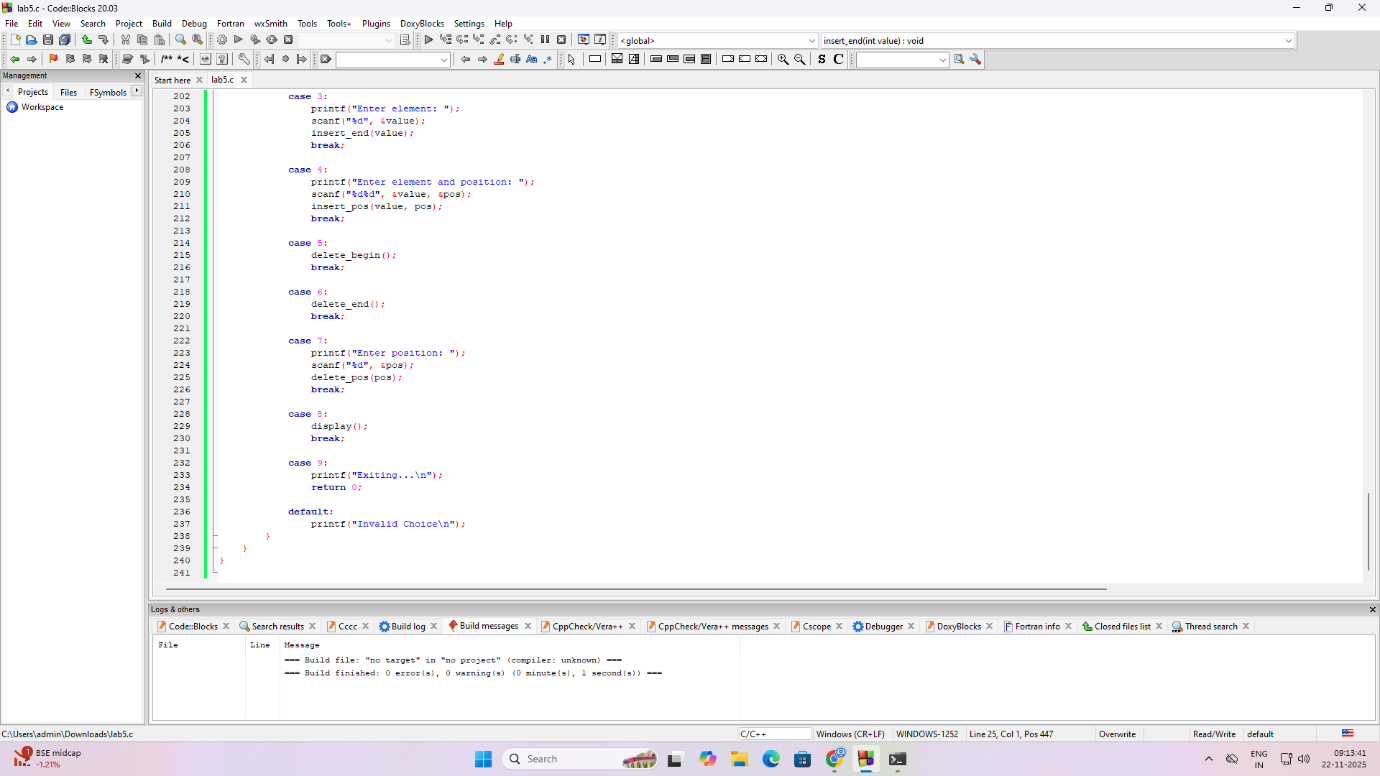
****

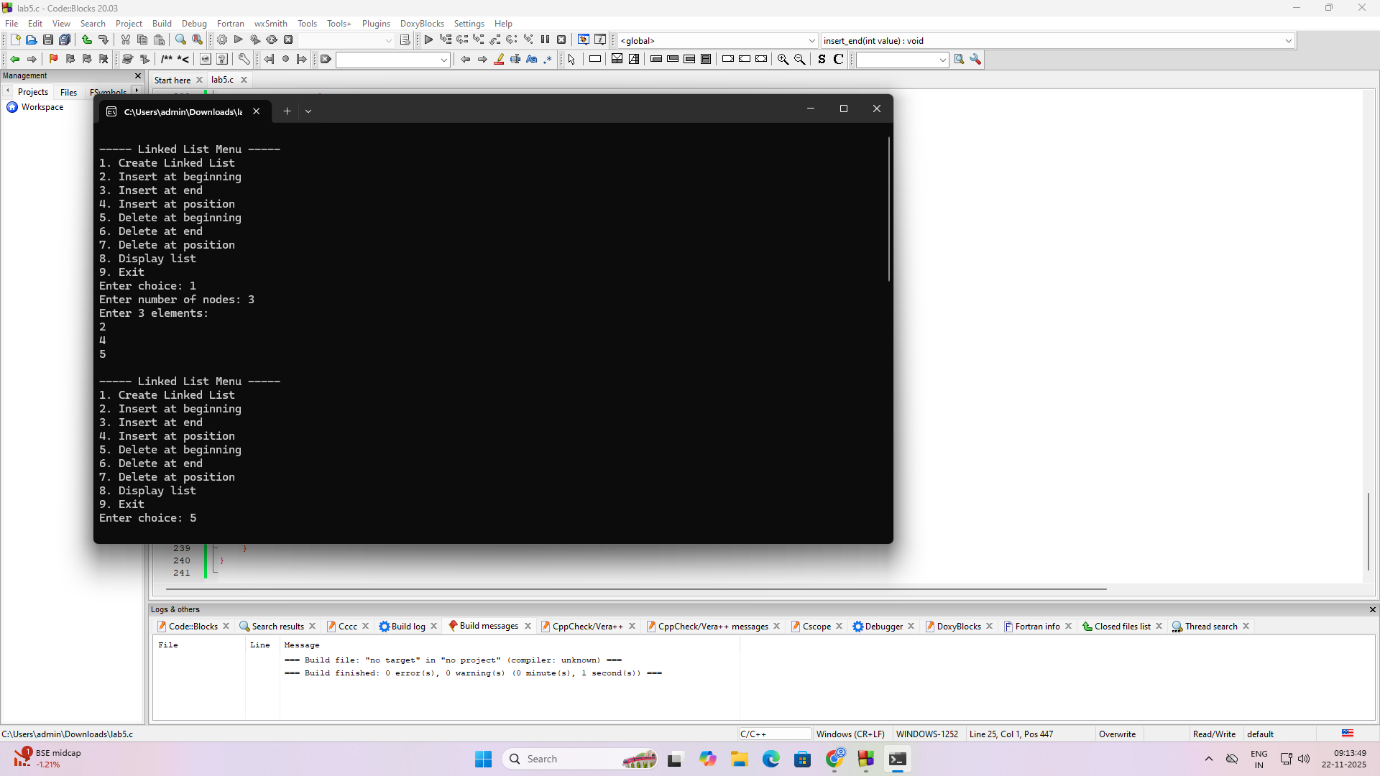
****

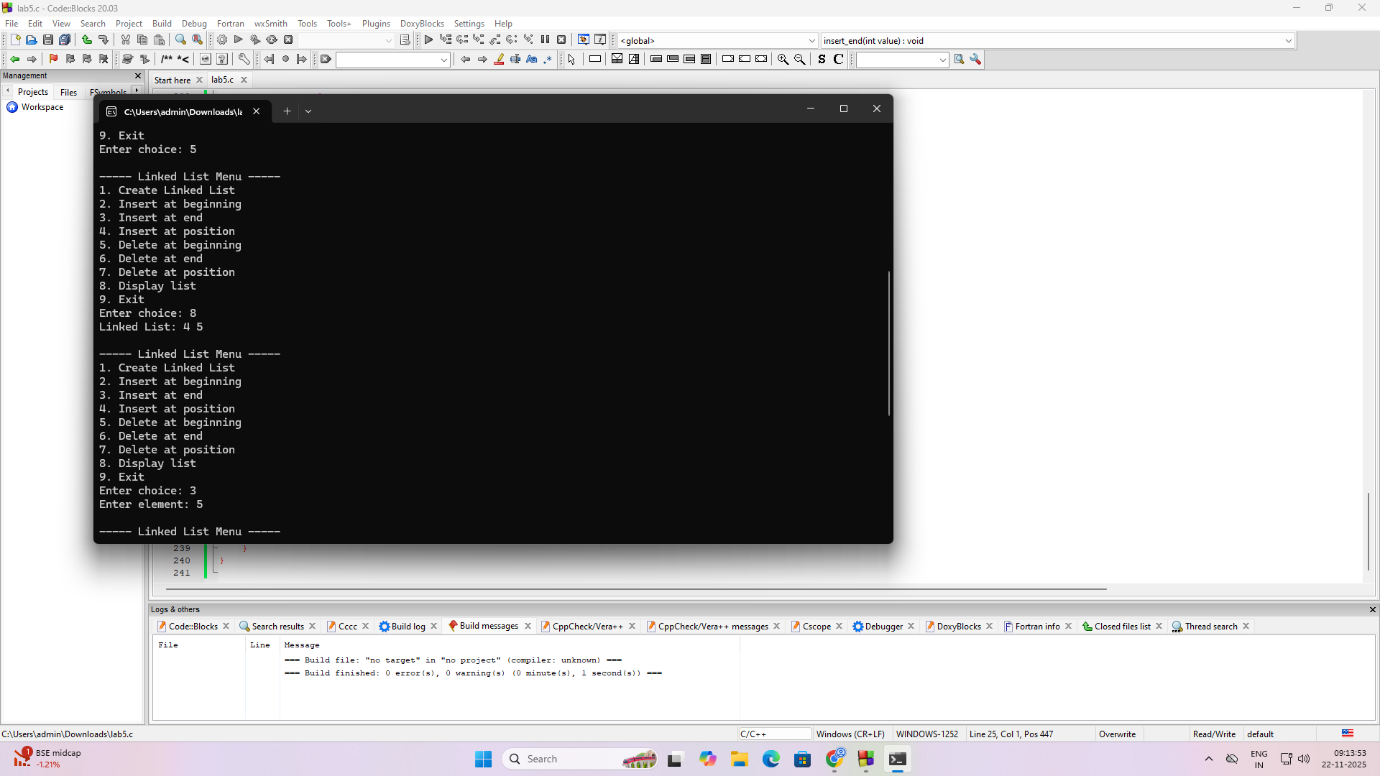
****

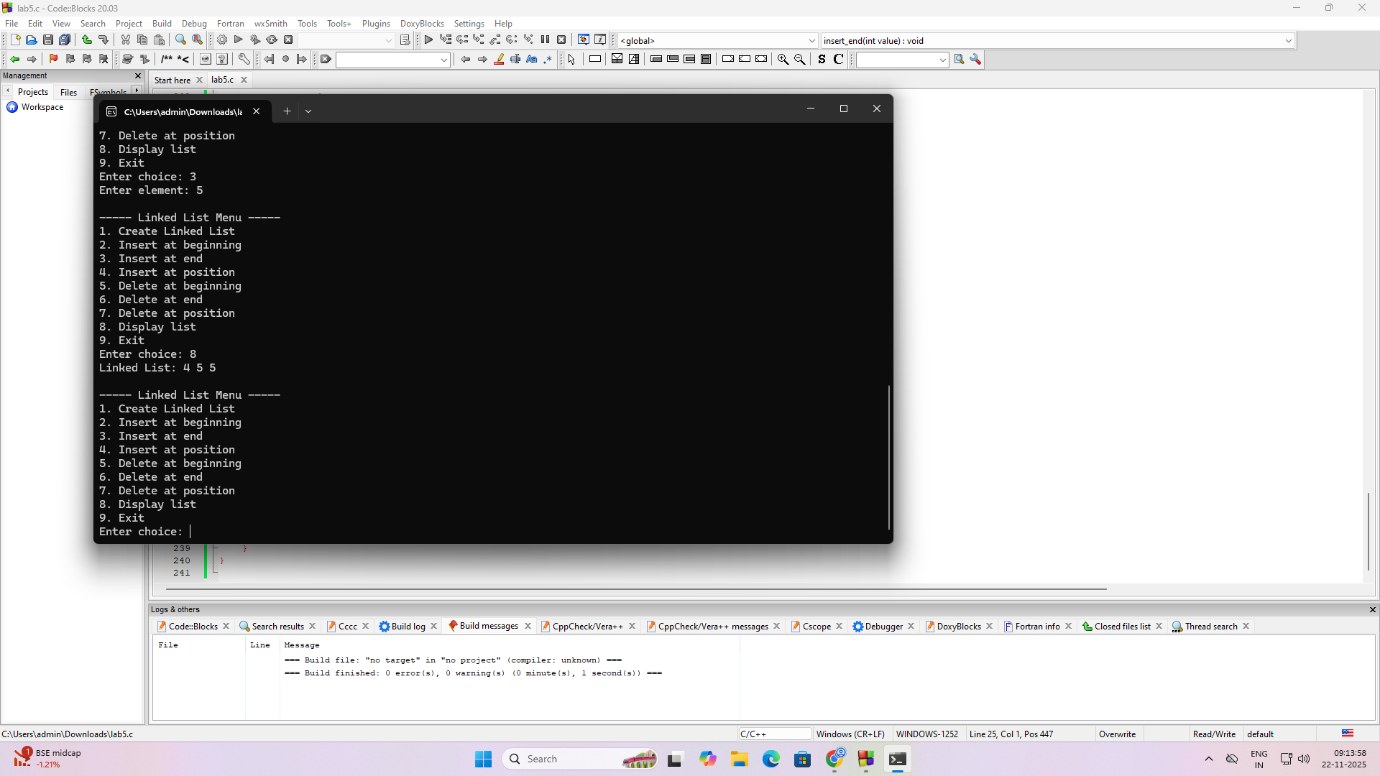
****

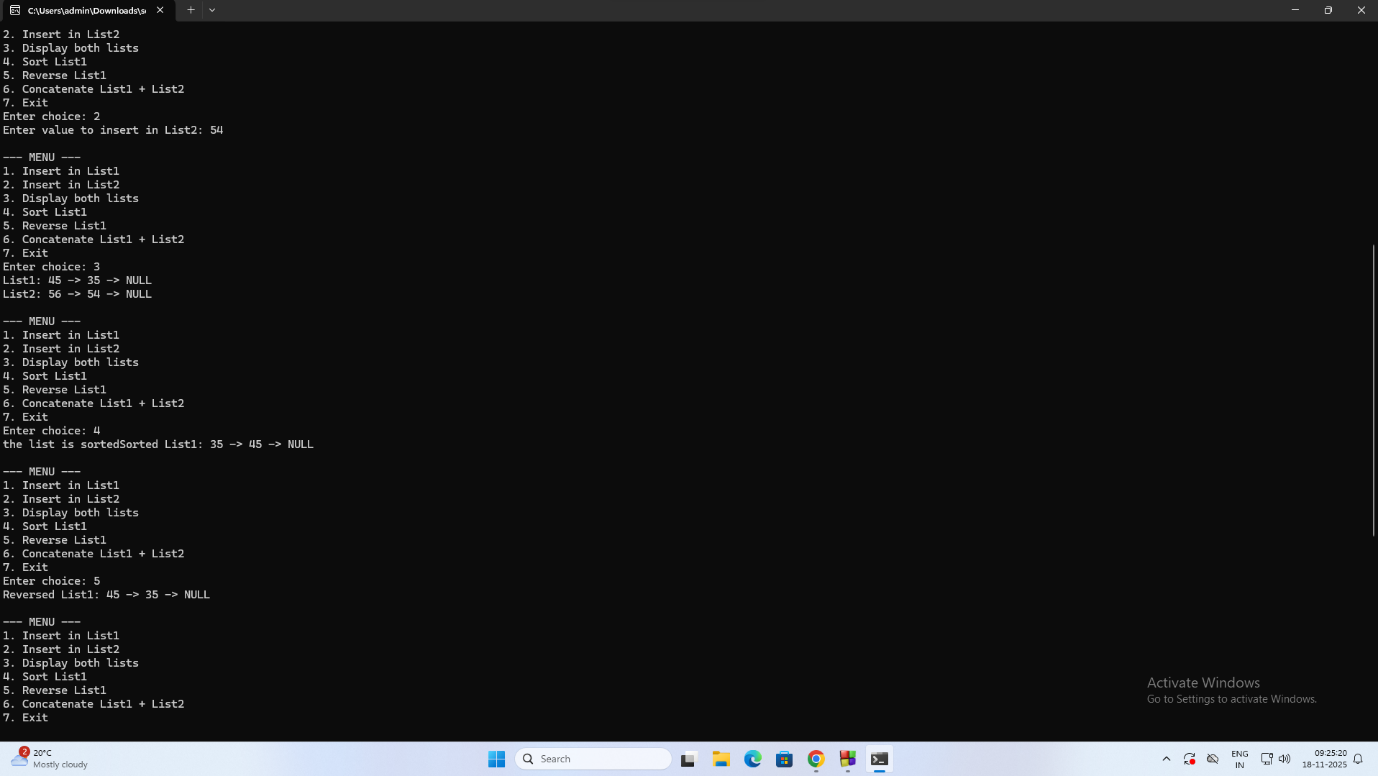
****

****

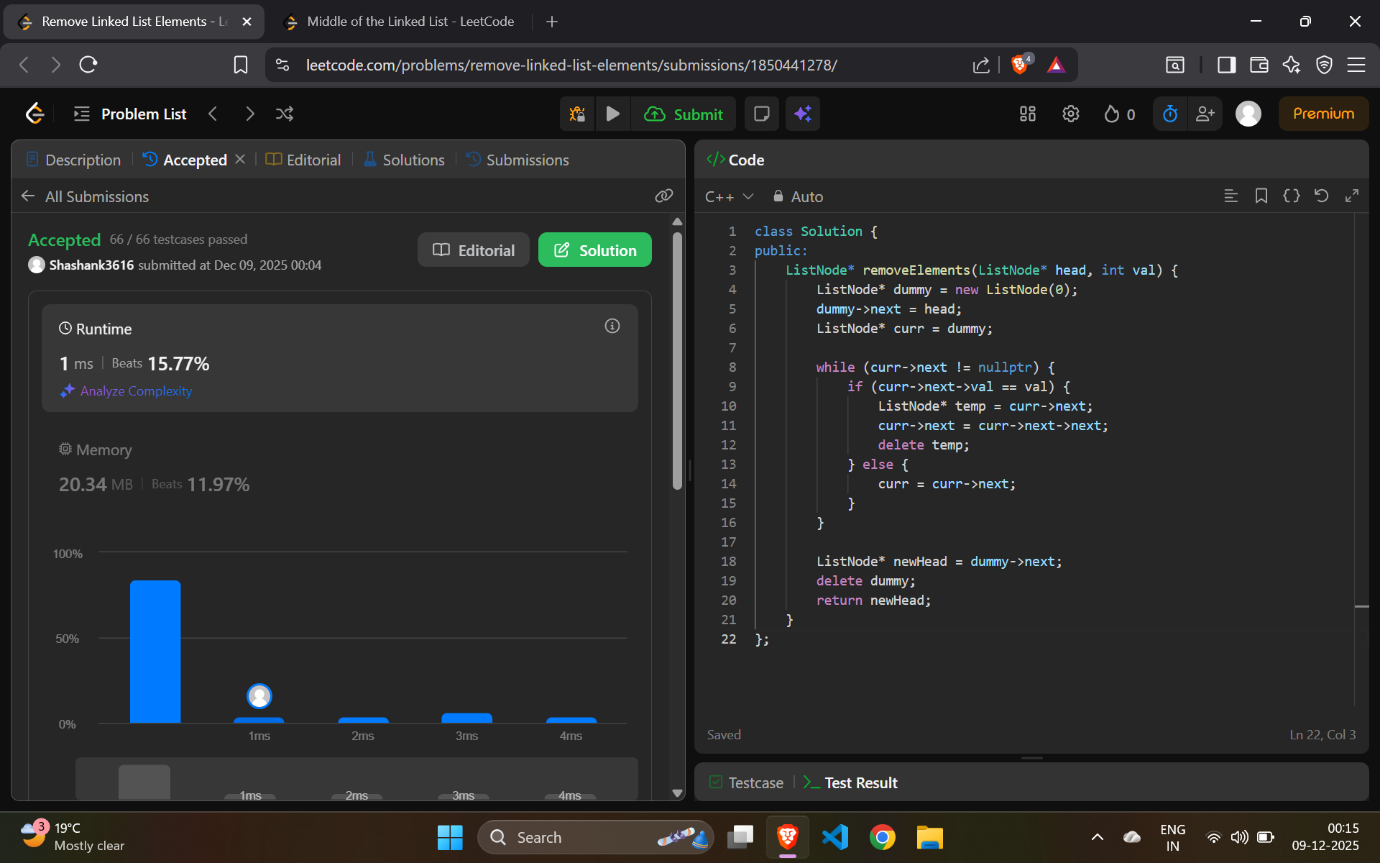
****

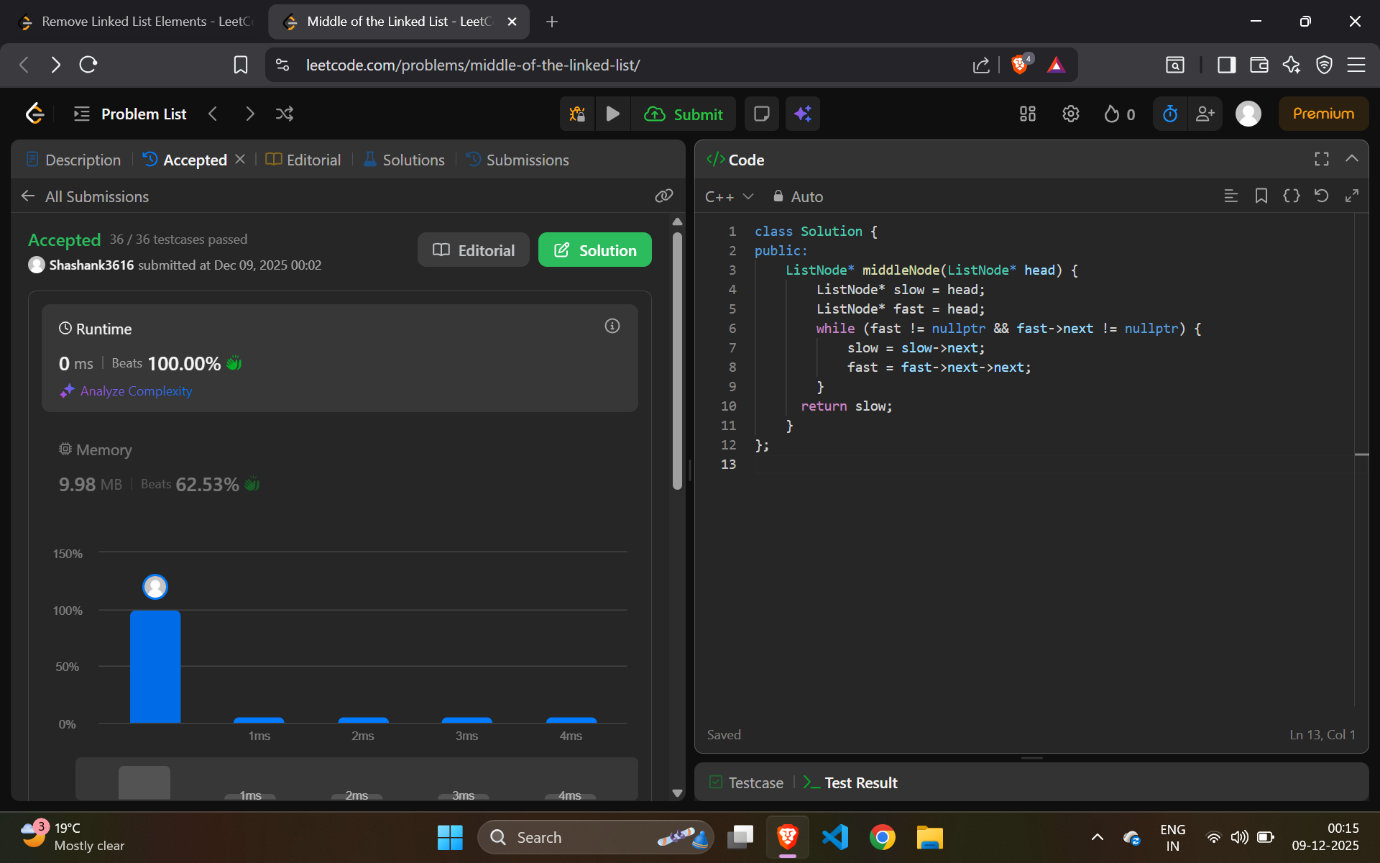
****

****

****

**4 b & 5b]** Program - Leetcode platform

****

****

**6 a) WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.**

**#include <stdio.h>**

**#include <stdlib.h>**

**struct Node {**

**int data;**

**struct Node\* next;**

**};**

**struct Node\* head = NULL;**

**void createList() {**

**int item;**

**struct Node \*newNode, \*temp;**

**printf("Enter elements to create list (-999 to stop): ");**

**scanf("%d", &item);**

**while (item != -999) {**

**newNode = (struct Node\*)malloc(sizeof(struct Node));**

**newNode->data = item;**

**newNode->next = NULL;**

**if (head == NULL) {**

**head = newNode;**

**} else {**

**temp = head;**

**while (temp->next != NULL)**

**temp = temp->next;**

**temp->next = newNode;**

**}**

**scanf("%d", &item);**

**}**

**}**

**struct Node\* createNode(int value) {**

**struct Node\* newNode = (struct Node\*) malloc(sizeof(struct Node));**

**newNode->data = value;**

**newNode->next = NULL;**

**return newNode;**

**}**

**// Display the linked list**

**void display(struct Node\* head) {**

**while (head != NULL) {**

**printf("%d -> ", head->data);**

**head = head->next;**

**}**

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

**}**

**void sortList(struct Node\* head) {**

**struct Node \*i, \*j;**

**int temp;**

**if (head == NULL)**

**return;**

**for (i = head; i->next != NULL; i = i->next) {**

**for (j = i->next; j != NULL; j = j->next) {**

**if (i->data > j->data) {**

**temp = i->data;**

**i->data = j->data;**

**j->data = temp;**

**}**

**}**

**}**

**}**

**void reverseList(struct Node\*\* head) {**

**struct Node\* prev = NULL;**

**struct Node\* curr = \*head;**

**struct Node\* next = NULL;**

**while (curr != NULL) {**

**next = curr->next;**

**curr->next = prev;**

**prev = curr;**

**curr = next;**

**}**

**\*head = prev;**

**}**

**// Concatenate two linked lists**

**struct Node\* concatenate(struct Node\* head1, struct Node\* head2) {**

**if (head1 == NULL) return head2;**

**if (head2 == NULL) return head1;**

**struct Node\* temp = head1;**

**while (temp->next != NULL)**

**temp = temp->next;**

**temp->next = head2;**

**return head1;**

**}**

**int main() {**

**int choice;**

**struct Node \*list2 = NULL;**

**while (1) {**

**printf("\n===== LINKED LIST MENU =====\n");**

**printf("1. Create List\n");**

**printf("2. Display List\n");**

**printf("3. Sort List\n");**

**printf("4. Reverse List\n");**

**printf("5. Concatenate Two Lists\n");**

**printf("6. Exit\n");**

**printf("Enter your choice: ");**

**scanf("%d", &choice);**

**switch (choice) {**

**case 1:**

**head = NULL;**

**createList();**

**break;**

**case 2:**

**printf("Linked List: ");**

**display(head);**

**break;**

**case 3:**

**sortList(head);**

**printf("List After Sorting: ");**

**display(head);**

**break;**

**case 4:**

**reverseList(&head);**

**printf("List After Reversing: ");**

**display(head);**

**break;**

**case 5:**

**printf("Create second list for concatenation:\n");**

**list2 = NULL;**

**head = head;**

**createList();**

**break;**

**case 6:**

**printf("Exiting...\n");**

**exit(0);**

**default:**

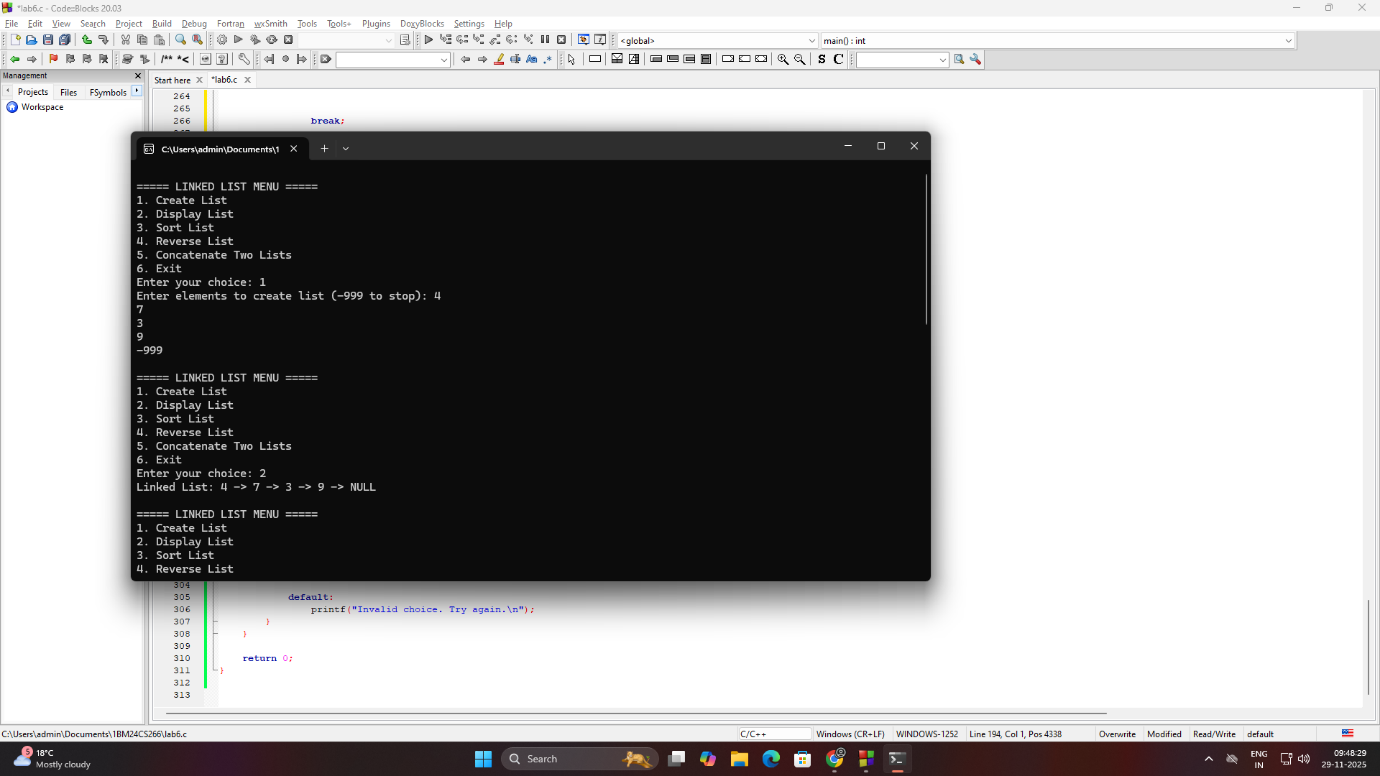
**printf("Invalid choice. Try again.\n");**

**}**

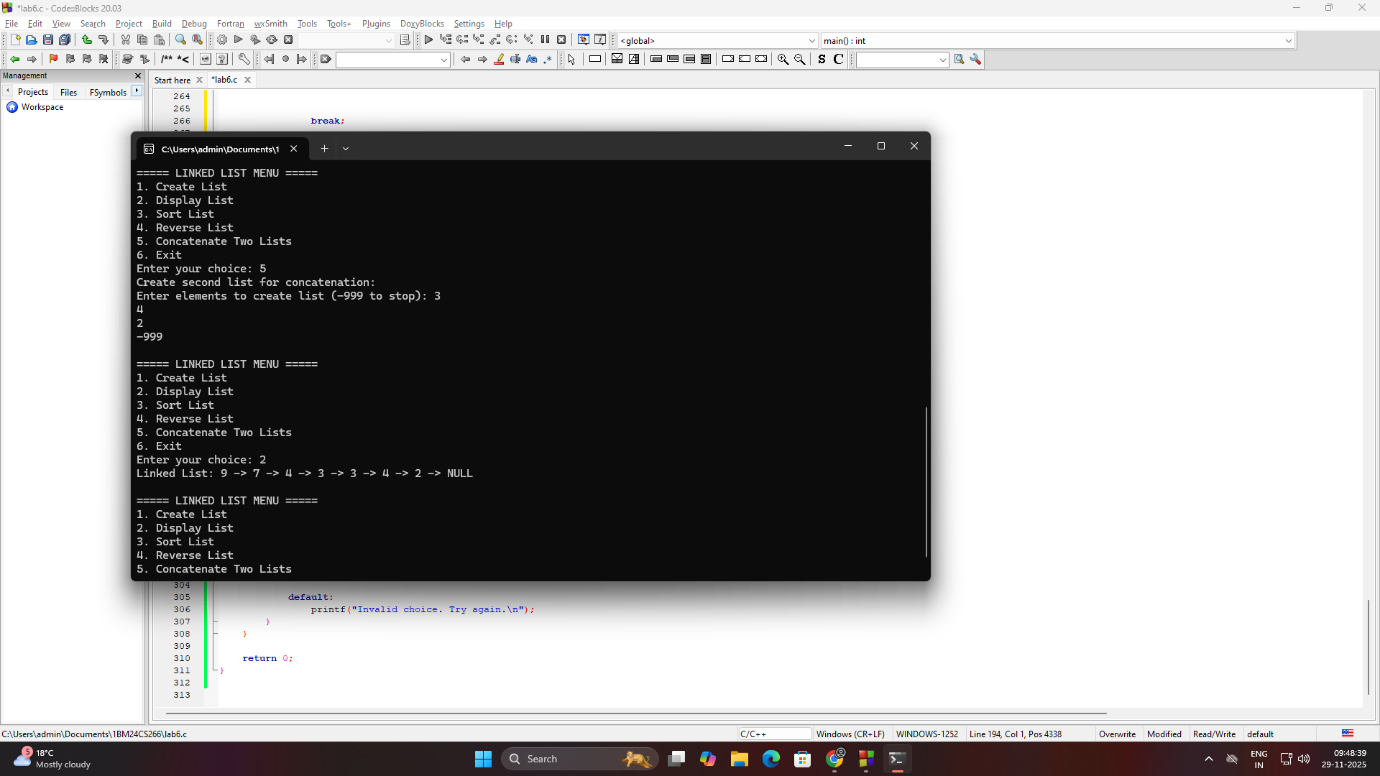
**}**

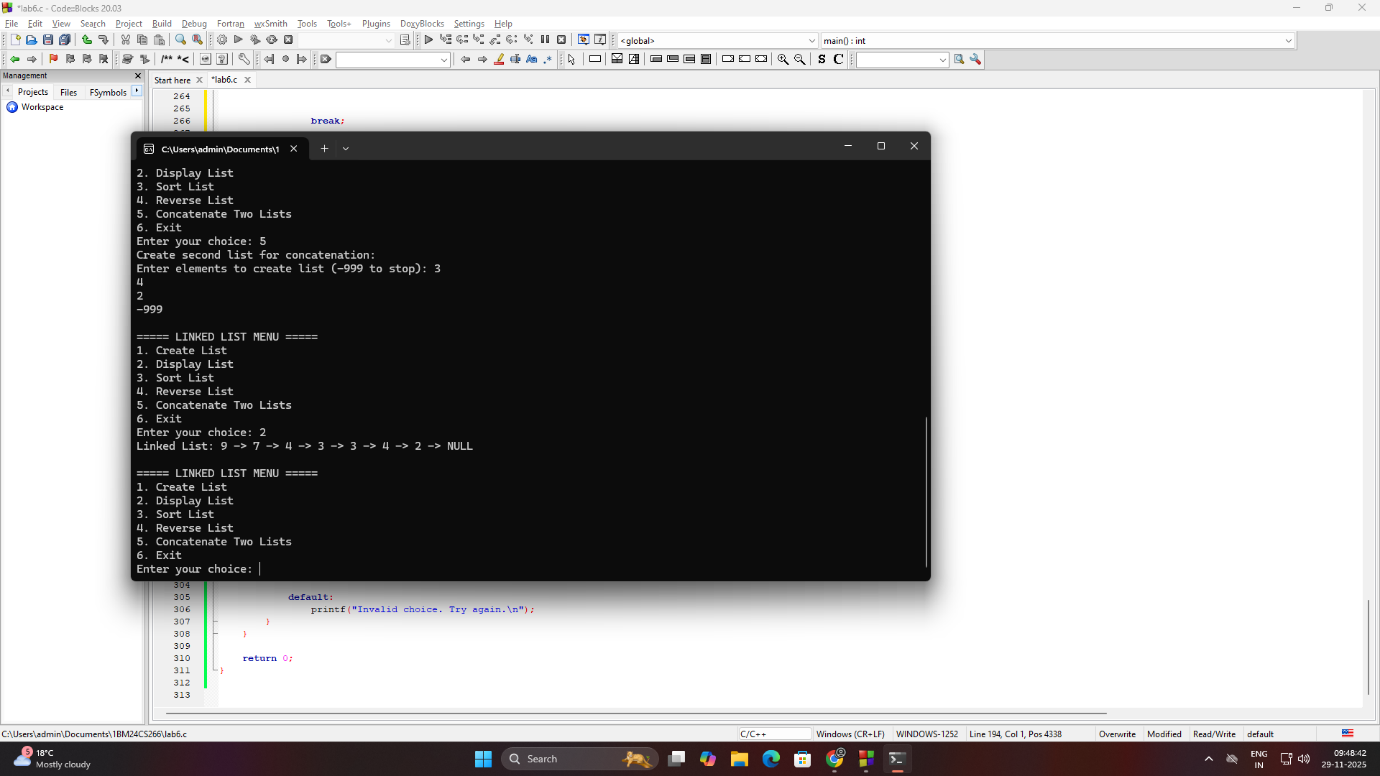
**return 0;**

**}**

****

****

****

****

**b) WAP to Implement Single Link List to simulate Stack & Queue Operations.**

**QUEUE**

**#include <stdio.h>**

**#include <stdlib.h>**

**struct Node {**

**int data;**

**struct Node\* next;**

**};**

**struct Node\* head = NULL;**

**void insertAtEnd(int item) {**

**struct Node \*newNode = (struct Node\*)malloc(sizeof(struct Node));**

**struct Node \*temp;**

**newNode->data = item;**

**newNode->next = NULL;**

**if (head == NULL) {**

**head = newNode;**

**return;**

**}**

**temp = head;**

**while (temp->next != NULL)**

**temp = temp->next;**

**temp->next = newNode;**

**}**

**void deleteFirst() {**

**if (head == NULL) {**

**printf("List is empty!\n");**

**return;**

**}**

**struct Node\* temp = head;**

**head = head->next;**

**printf("Deleted: %d\n", temp->data);**

**free(temp);**

**}**

**void display() {**

**struct Node \*temp = head;**

**if (head == NULL) {**

**printf("List is empty.\n");**

**return;**

**}**

**printf("Linked List: ");**

**while (temp != NULL) {**

**printf("%d -> ", temp->data);**

**temp = temp->next;**

**}**

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

**}**

**int main() {**

**int choice, item;**

**while (1) {**

**printf("\n===== QUEUE MENU (Using Linked List) =====\n");**

**printf("1. Push (Insert)\n");**

**printf("2. Pop (Delete last)\n");**

**printf("3. Display Stack\n");**

**printf("4. Exit\n");**

**printf("Enter your choice: ");**

**scanf("%d", &choice);**

**switch (choice) {**

**case 1:**

**printf("Enter value to push: ");**

**scanf("%d", &item);**

**insertAtEnd(item);**

**break;**

**case 2:**

**deleteFirst();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**printf("Exiting...\n");**

**exit(0);**

**default:**

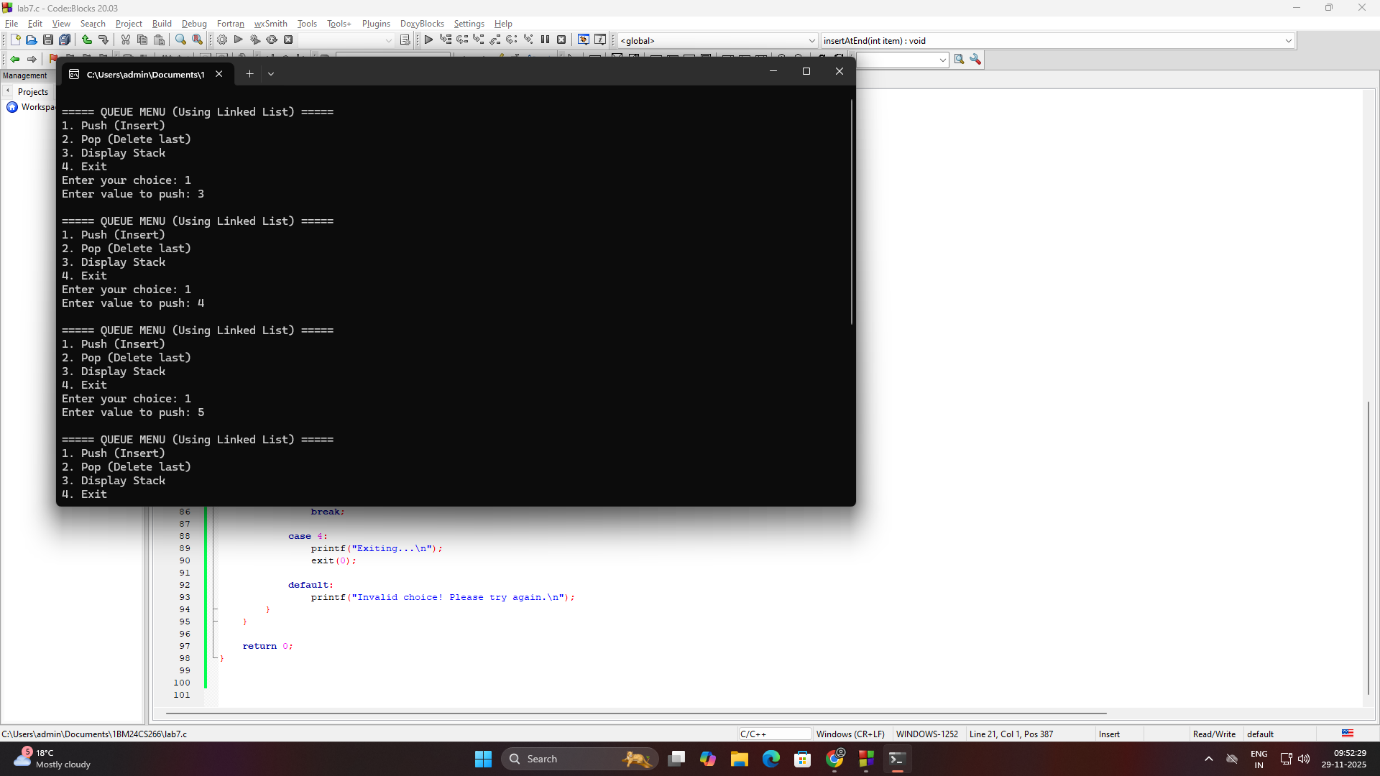
**printf("Invalid choice! Please try again.\n");**

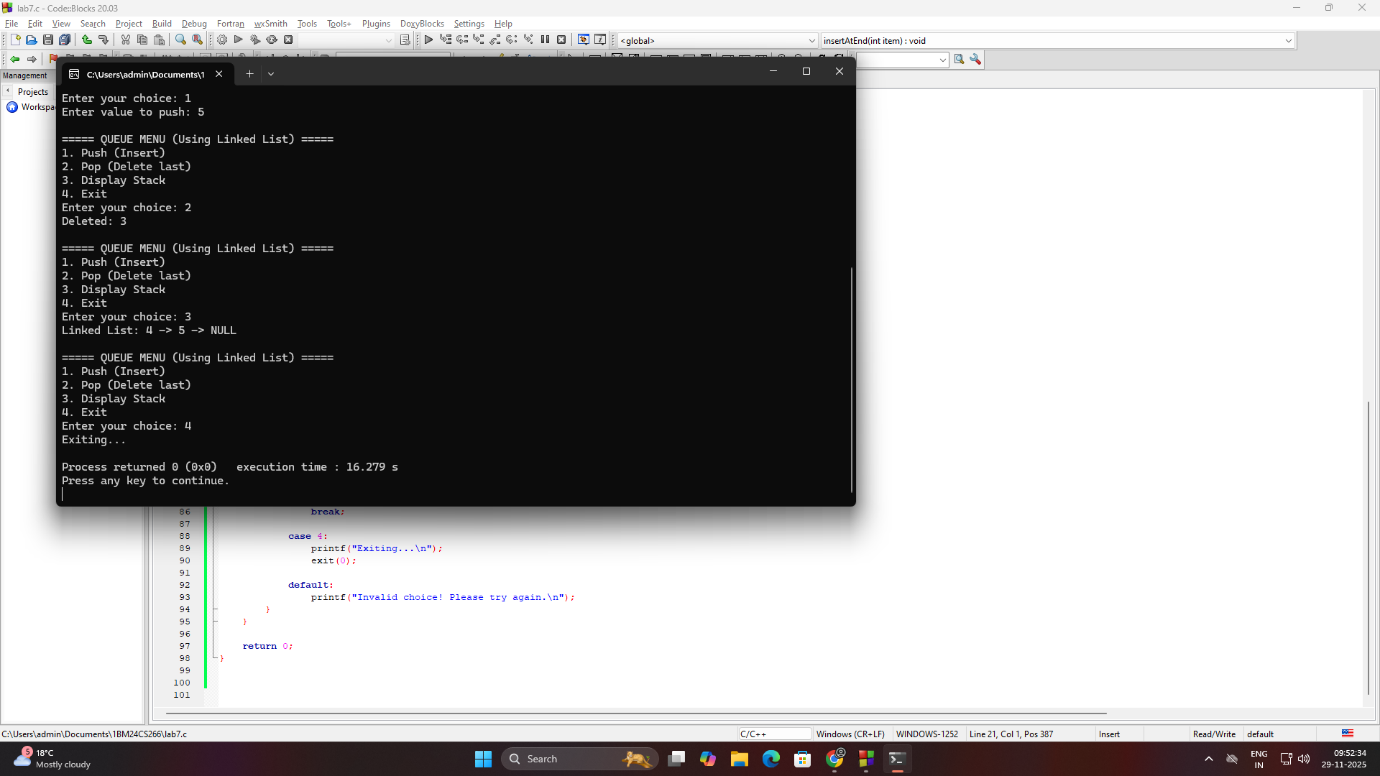
**}**

**}**

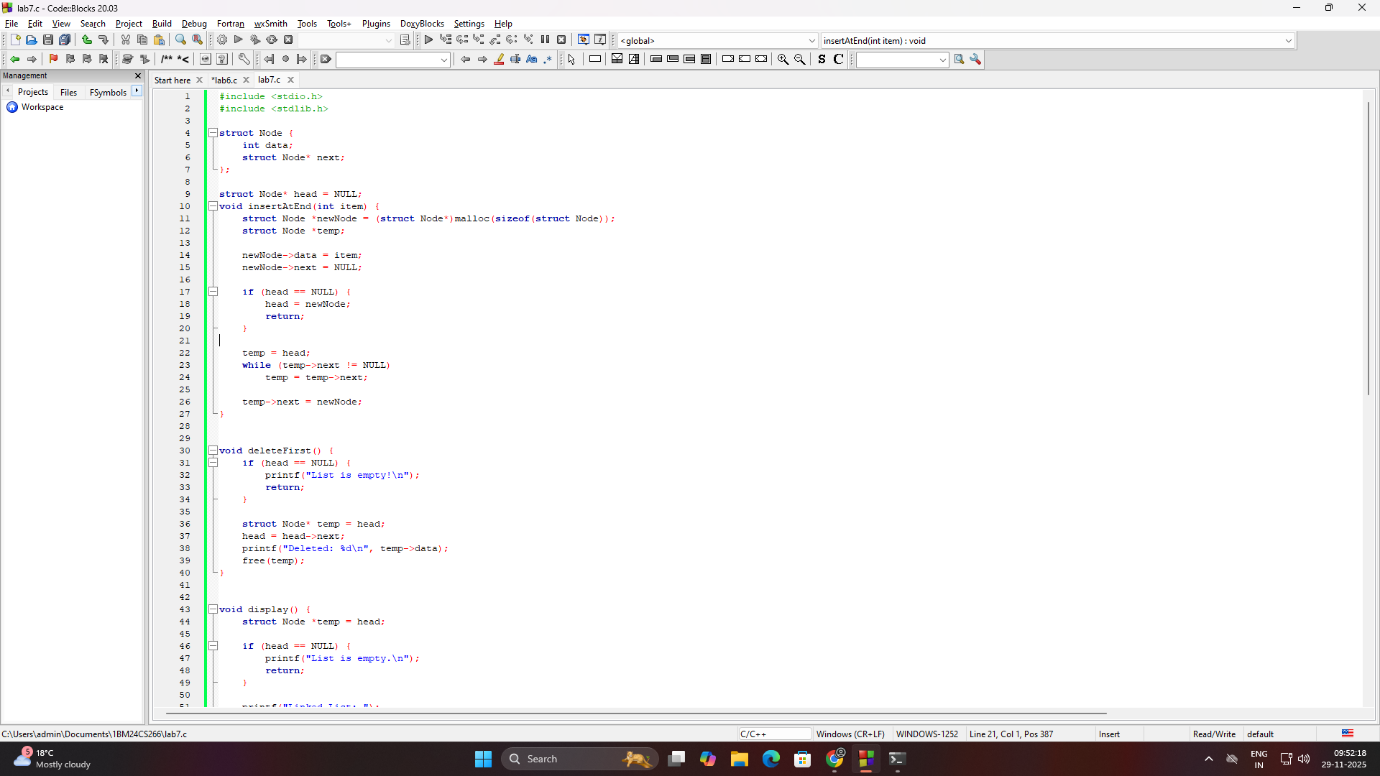
**return 0;**

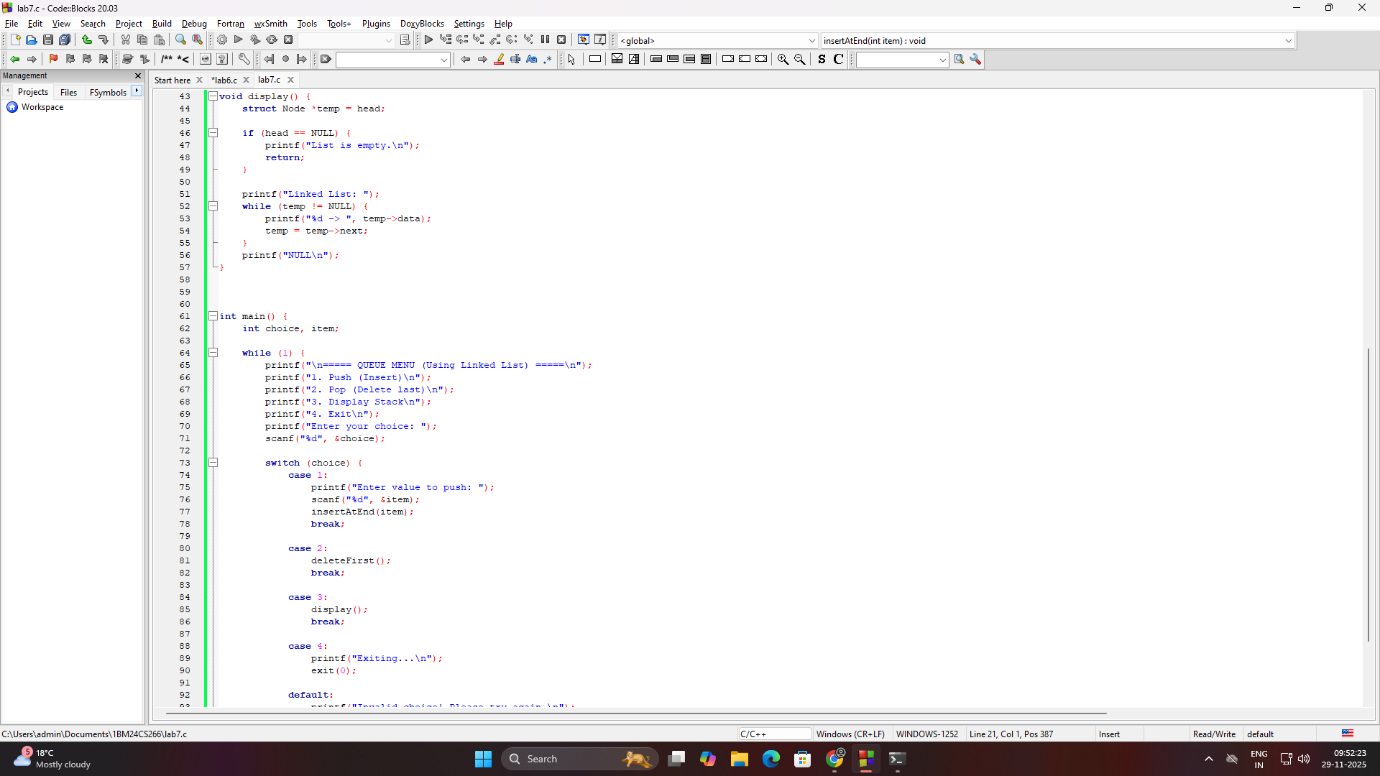
**}**

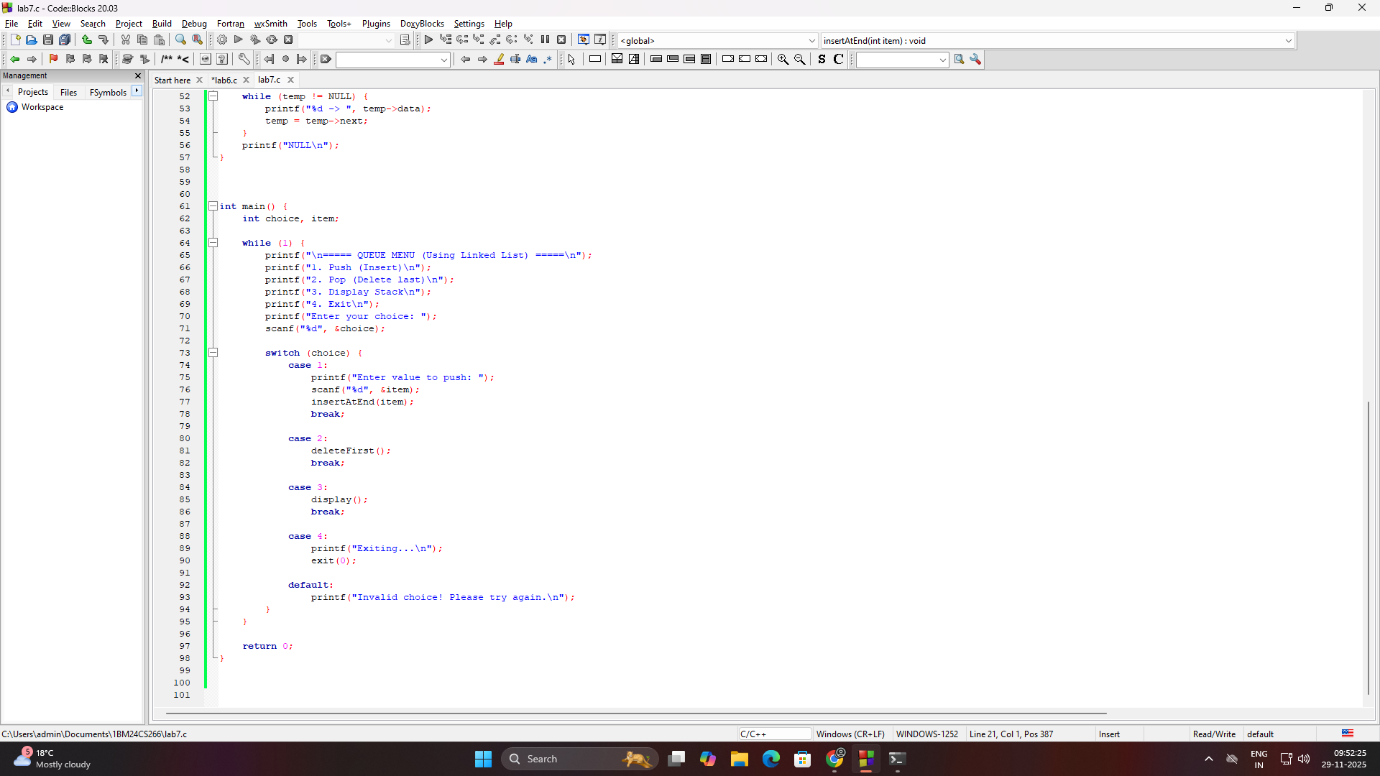
****

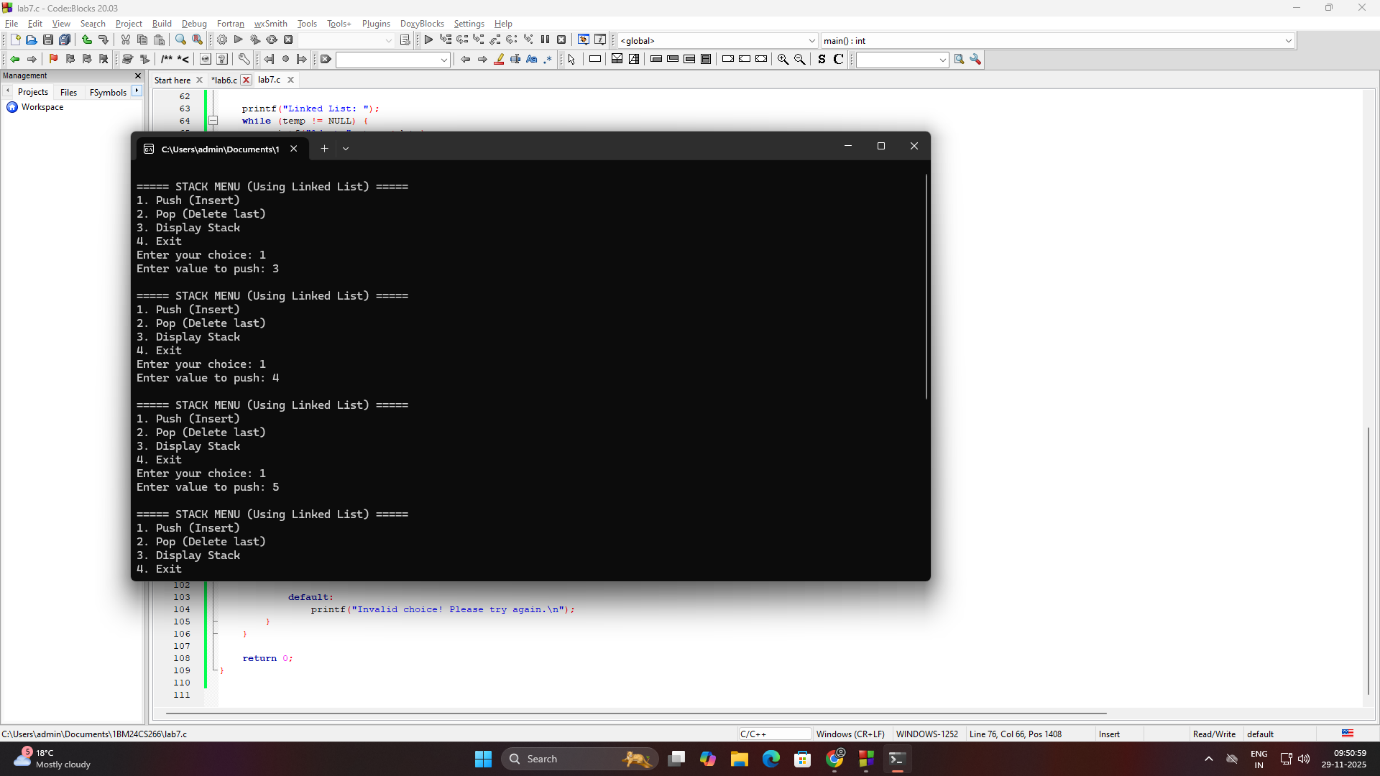
****

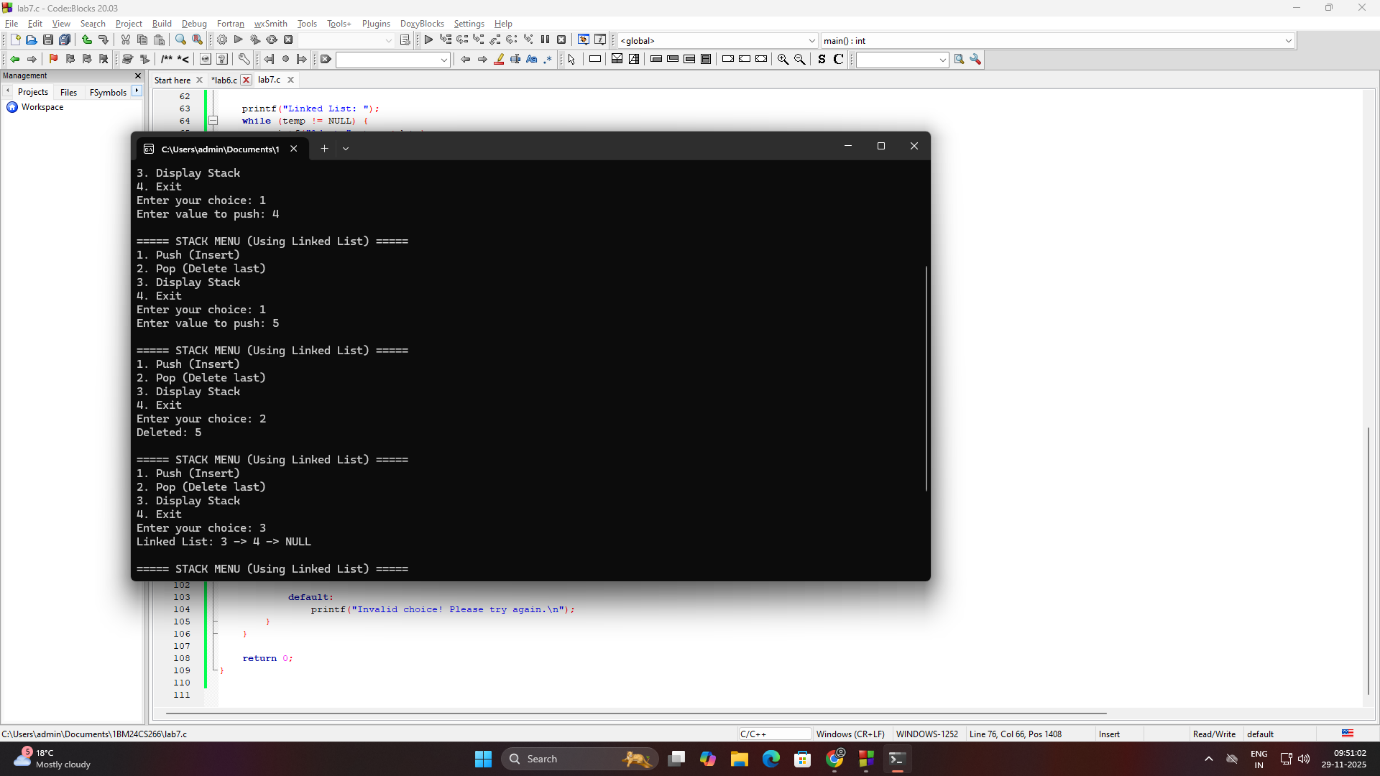
**STACK**

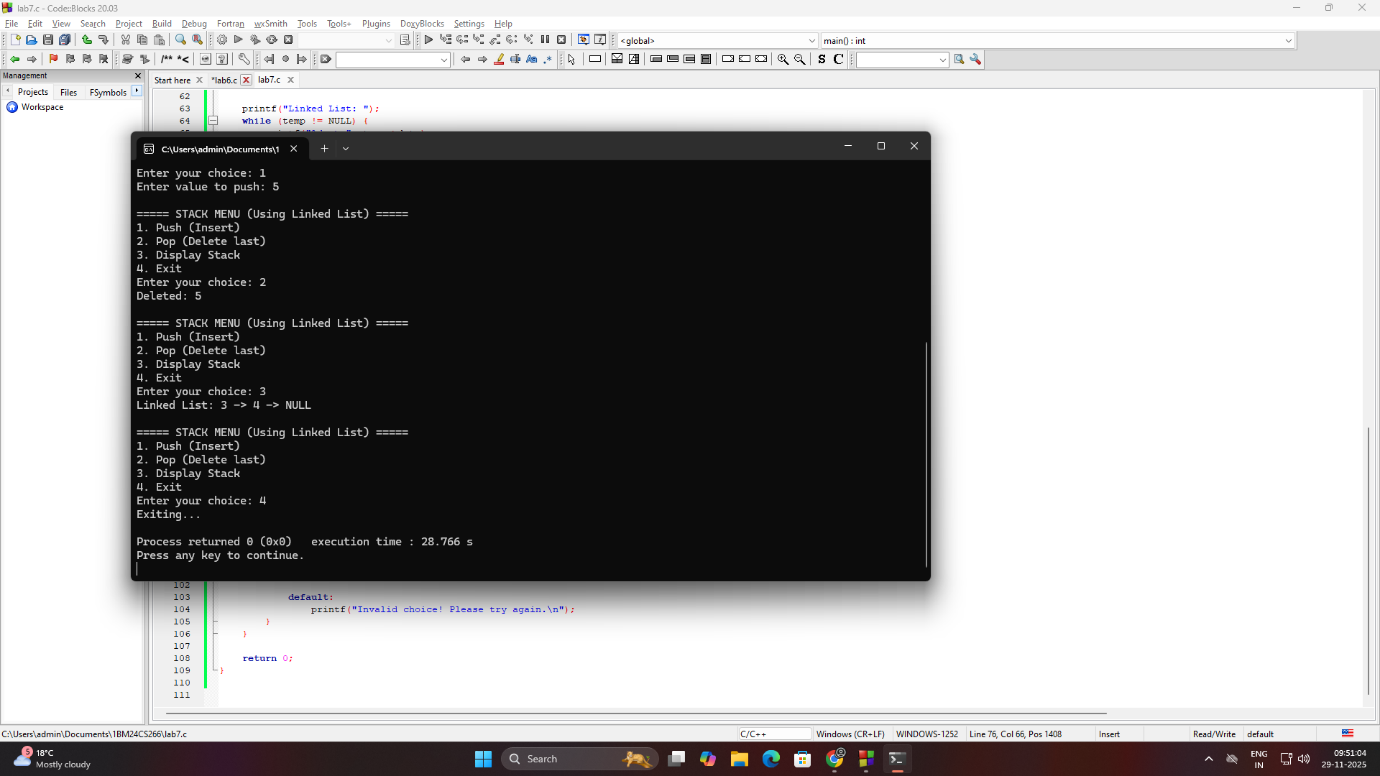
****

****

****

****

****

****