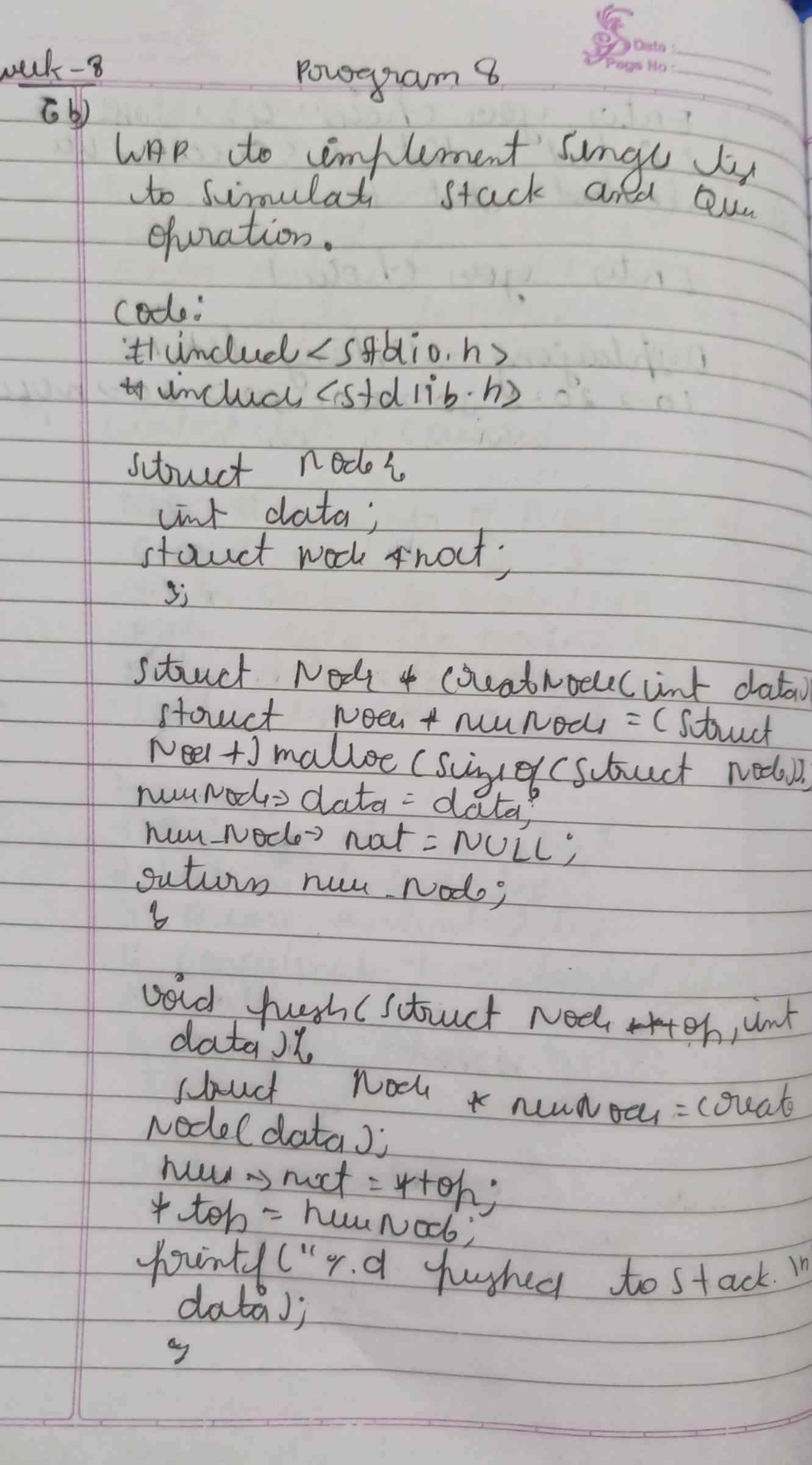
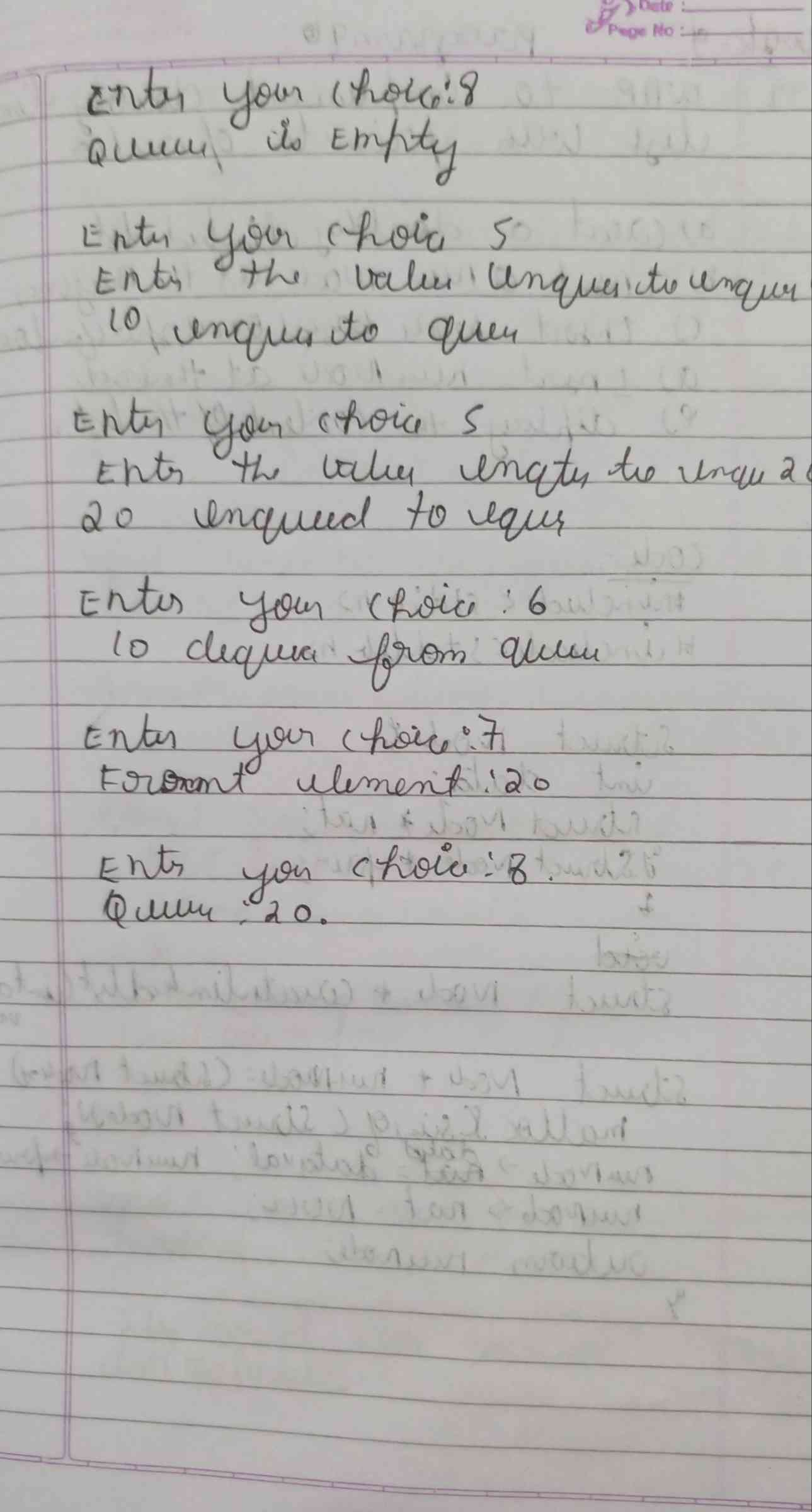
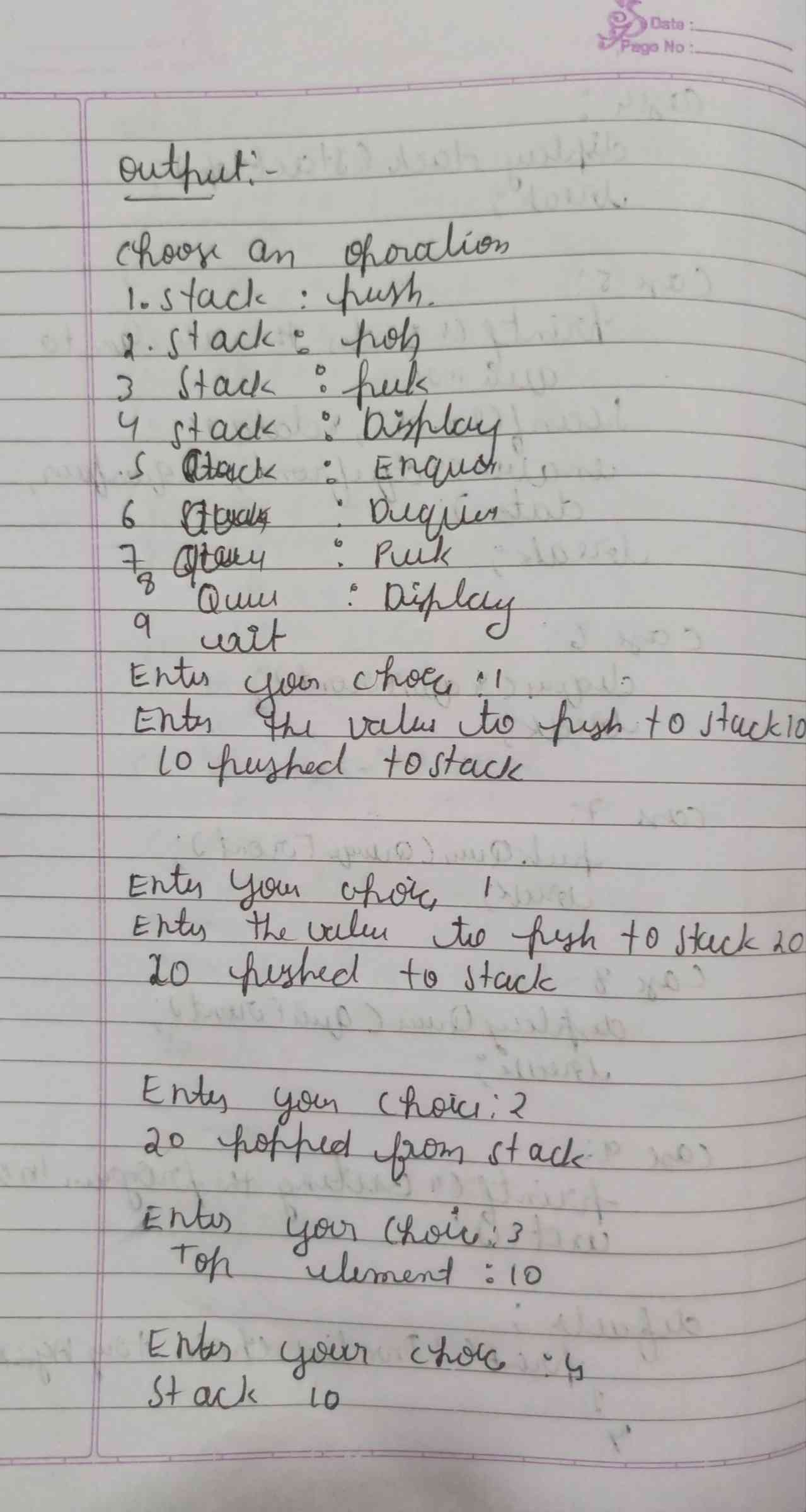
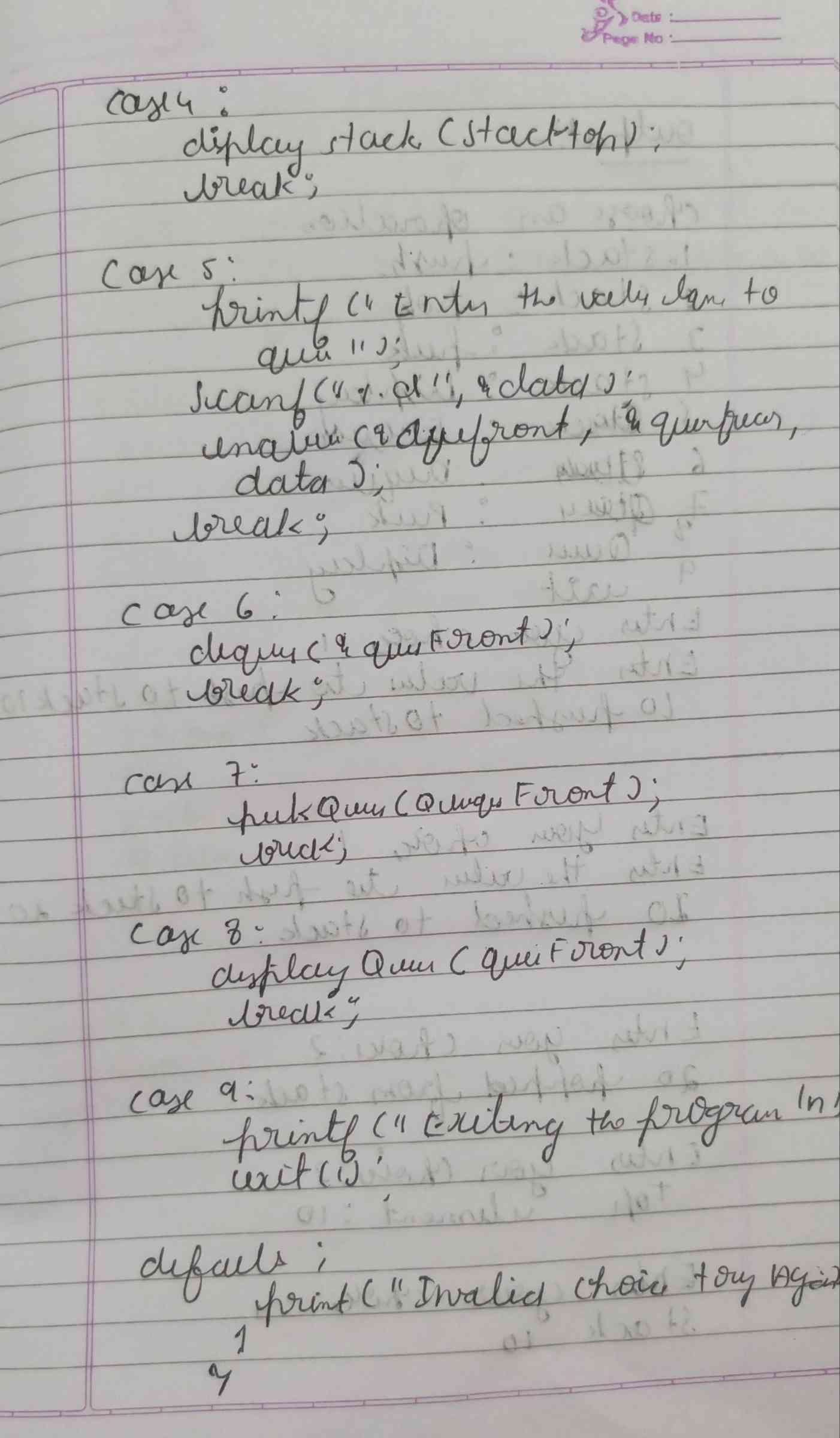
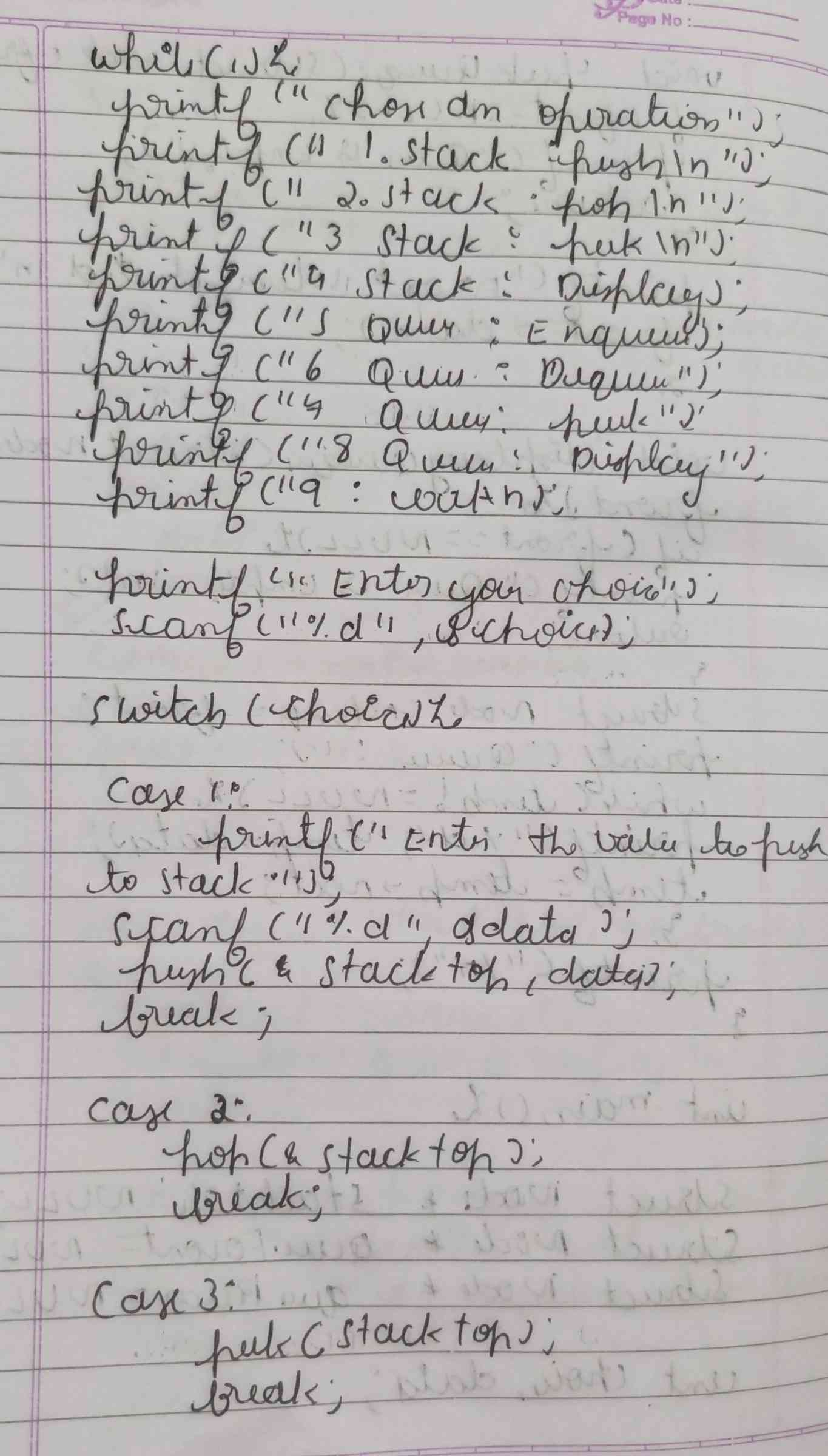
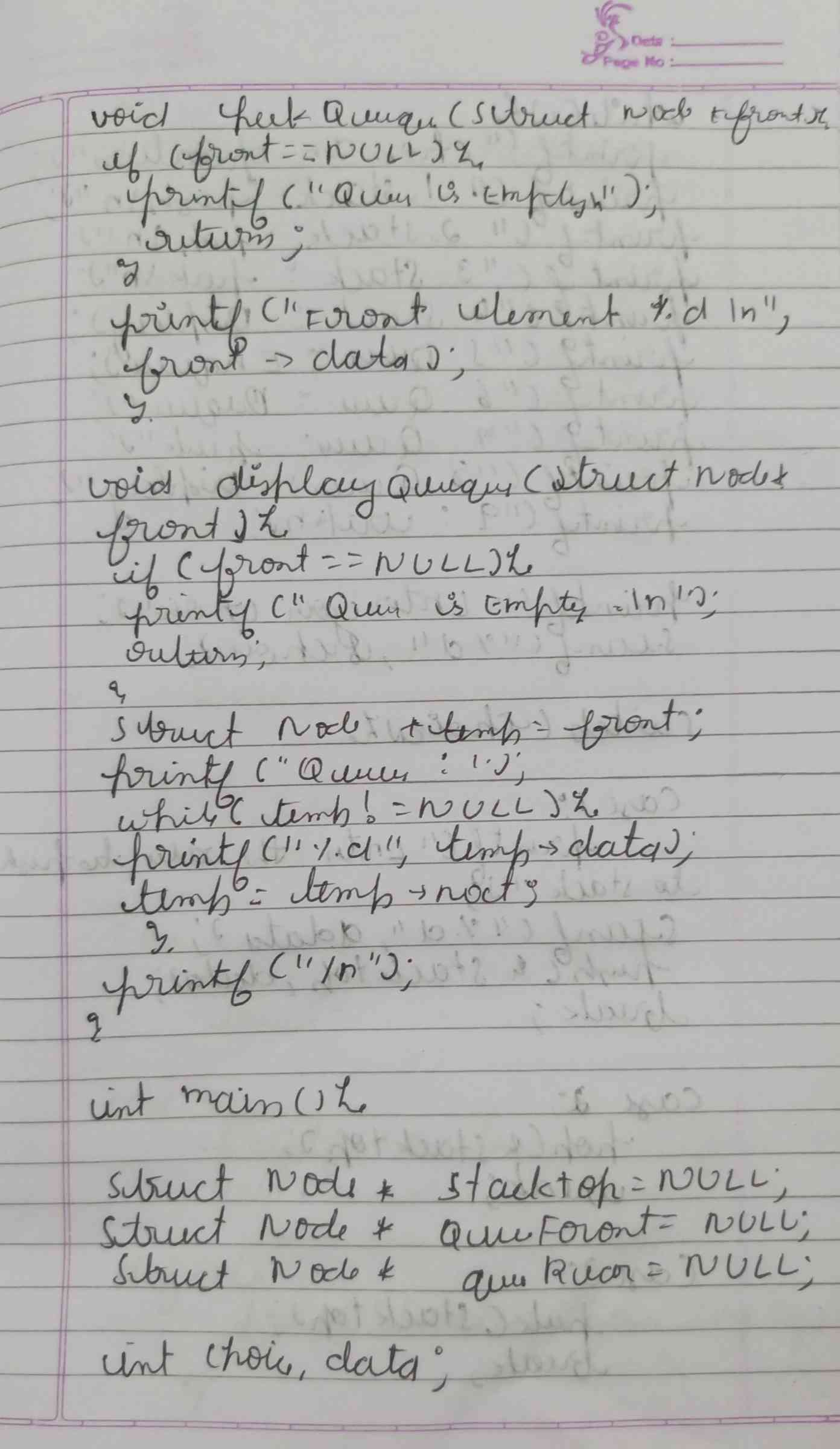
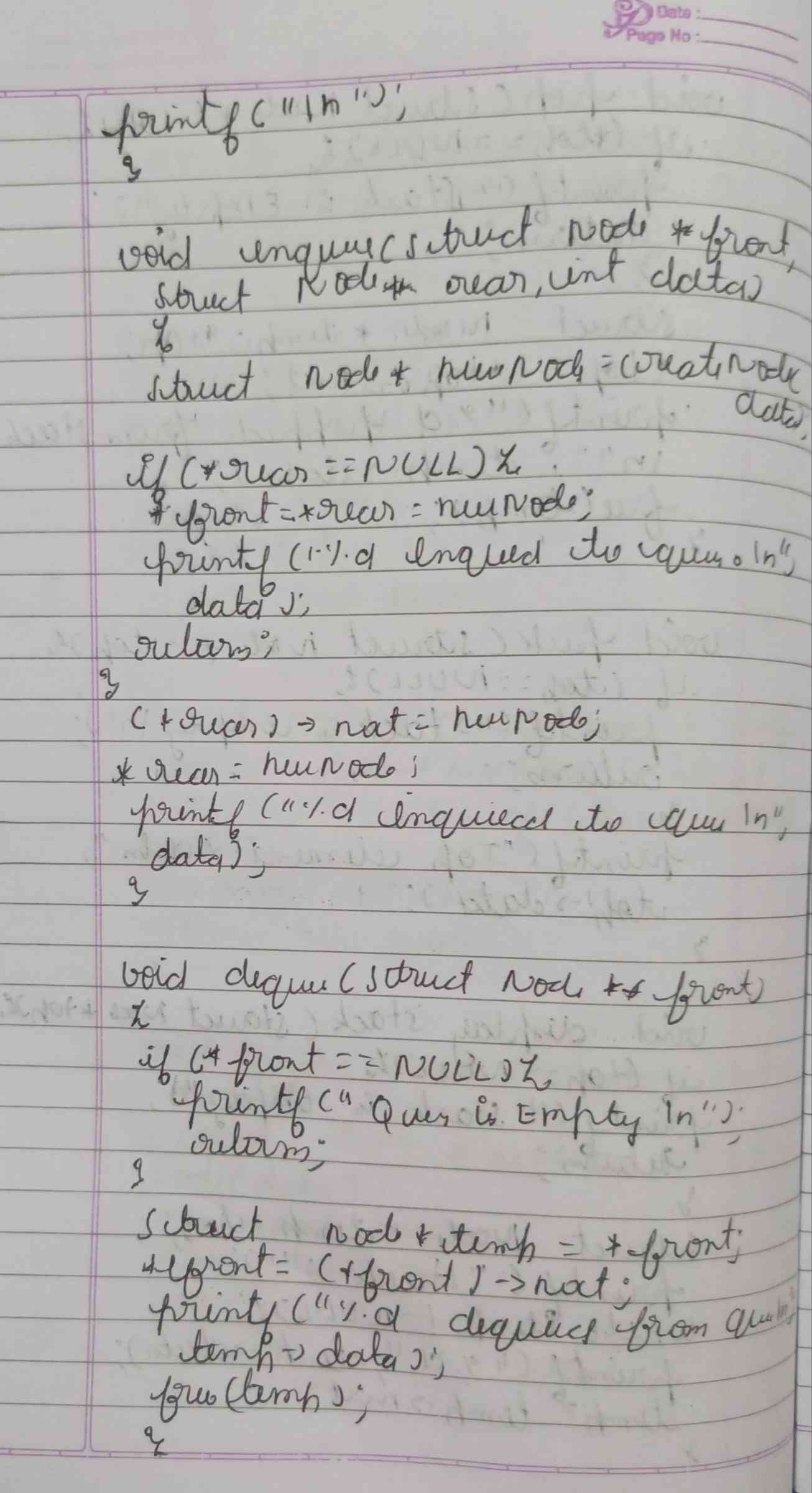
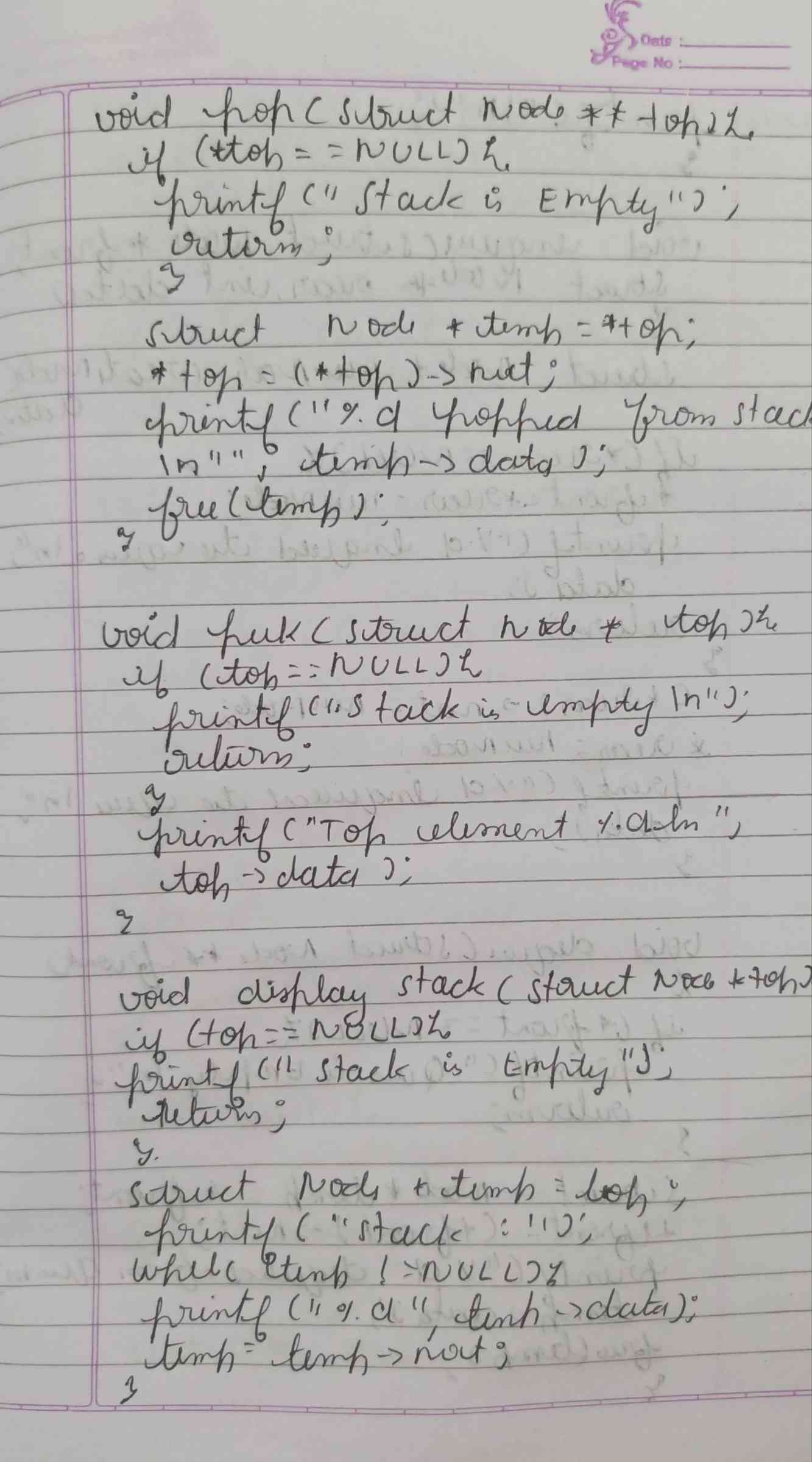
**LABORATORY PROGRAM – 8**

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

**OBSERVATION :**



****

**CODE :**

**#include <stdio.h>**

**#include <stdlib.h>**

**struct Node {**

**int data;**

**struct Node\* next;**

**};**

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

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

**newNode->data = data;**

**newNode->next = NULL;**

**return newNode;**

**}**

**void push(struct Node\*\* top, int data) {**

**struct Node\* newNode = createNode(data);**

**newNode->next = \*top;**

**\*top = newNode;**

**printf("%d pushed to stack.\n", data);**

**}**

**void pop(struct Node\*\* top) {**

**if (\*top == NULL) {**

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

**return;**

**}**

**struct Node\* temp = \*top;**

**\*top = (\*top)->next;**

**printf("%d popped from stack.\n", temp->data);**

**free(temp);**

**}**

**void peek(struct Node\* top) {**

**if (top == NULL) {**

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

**return;**

**}**

**printf("Top element: %d\n", top->data);**

**}**

**void displayStack(struct Node\* top) {**

**if (top == NULL) {**

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

**return;**

**}**

**struct Node\* temp = top;**

**printf("Stack: ");**

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

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

**temp = temp->next;**

**}**

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

**}**

**void enqueue(struct Node\*\* front, struct Node\*\* rear, int data) {**

**struct Node\* newNode = createNode(data);**

**if (\*rear == NULL) {**

**\*front = \*rear = newNode;**

**printf("%d enqueued to queue.\n", data);**

**return;**

**}**

**(\*rear)->next = newNode;**

**\*rear = newNode;**

**printf("%d enqueued to queue.\n", data);**

**}**

**void dequeue(struct Node\*\* front) {**

**if (\*front == NULL) {**

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

**return;**

**}**

**struct Node\* temp = \*front;**

**\*front = (\*front)->next;**

**printf("%d dequeued from queue.\n", temp->data);**

**free(temp);**

**}**

**void peekQueue(struct Node\* front) {**

**if (front == NULL) {**

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

**return;**

**}**

**printf("Front element: %d\n", front->data);**

**}**

**void displayQueue(struct Node\* front) {**

**if (front == NULL) {**

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

**return;**

**}**

**struct Node\* temp = front;**

**printf("Queue: ");**

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

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

**temp = temp->next;**

**}**

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

**}**

**int main() {**

**struct Node\* stackTop = NULL;**

**struct Node\* queueFront = NULL;**

**struct Node\* queueRear = NULL;**

**int choice, data;**

**while(1) {**

**printf("\nChoose an operation:\n");**

**printf("1. Stack: Push\n");**

**printf("2. Stack: Pop\n");**

**printf("3. Stack: Peek\n");**

**printf("4. Stack: Display\n");**

**printf("5. Queue: Enqueue\n");**

**printf("6. Queue: Dequeue\n");**

**printf("7. Queue: Peek\n");**

**printf("8. Queue: Display\n");**

**printf("9. Exit\n");**

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

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

**switch (choice) {**

**case 1:**

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

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

**push(&stackTop, data);**

**break;**

**case 2:**

**pop(&stackTop);**

**break;**

**case 3:**

**peek(stackTop);**

**break;**

**case 4:**

**displayStack(stackTop);**

**break;**

**case 5:**

**printf("Enter value to enqueue to queue: ");**

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

**enqueue(&queueFront, &queueRear, data);**

**break;**

**case 6:**

**dequeue(&queueFront);**

**break;**

**case 7:**

**peekQueue(queueFront);**

**break;**

**case 8:**

**displayQueue(queueFront);**

**break;**

**case 9:**

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

**exit(1);**

**default:**

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

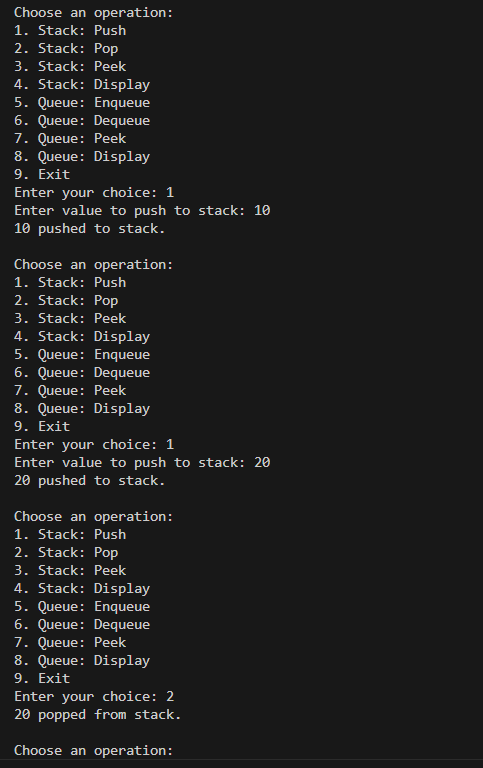
**}**

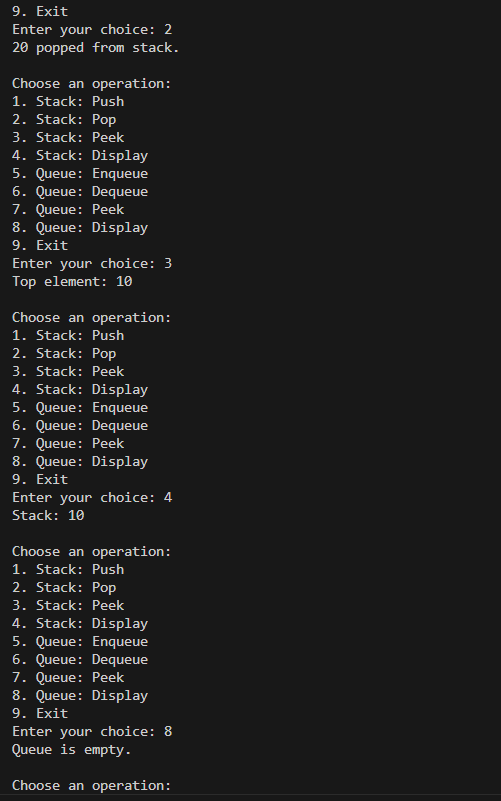
**}**

**return 0;**

**}**

**OUTPUT :**

****

****