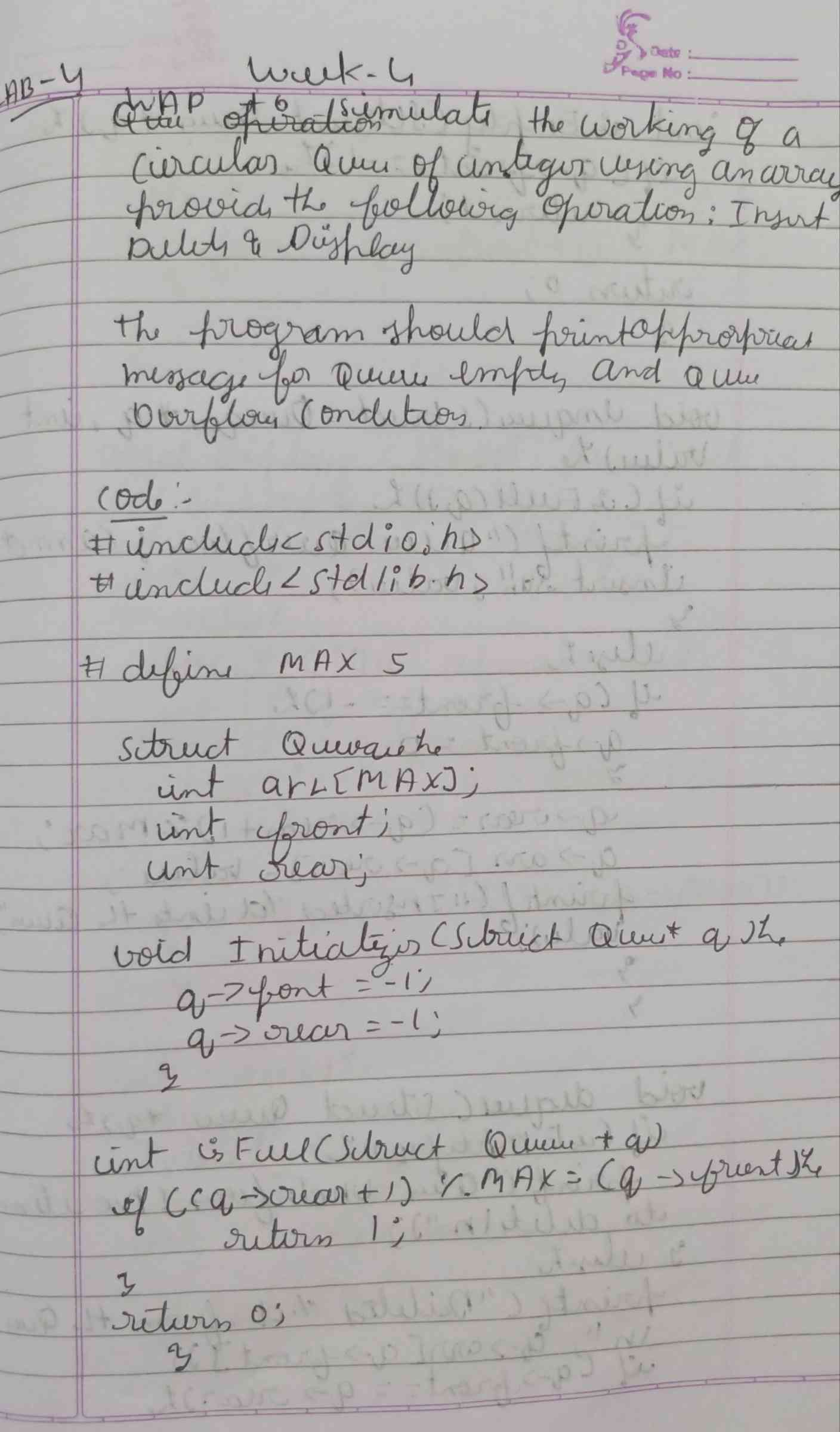
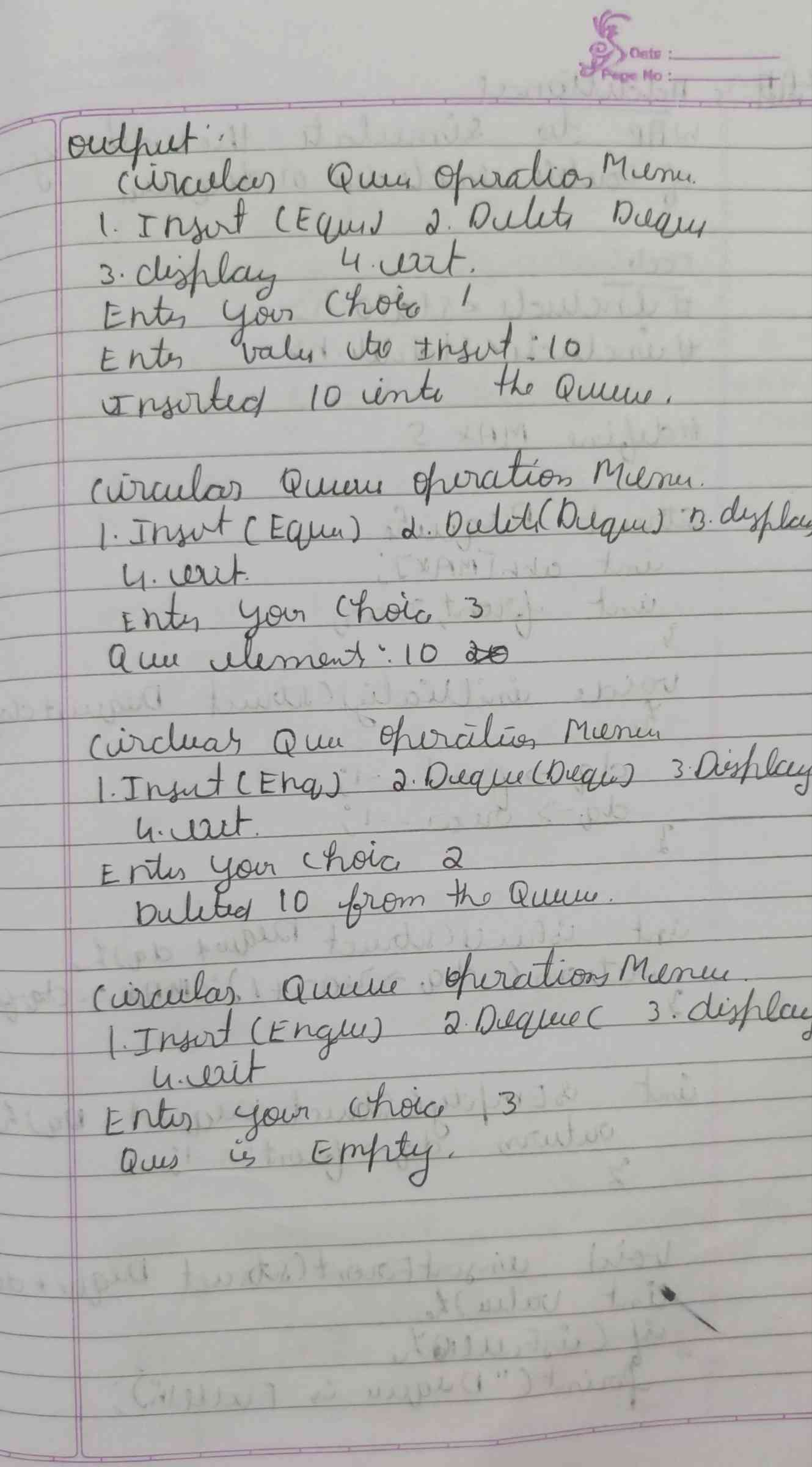
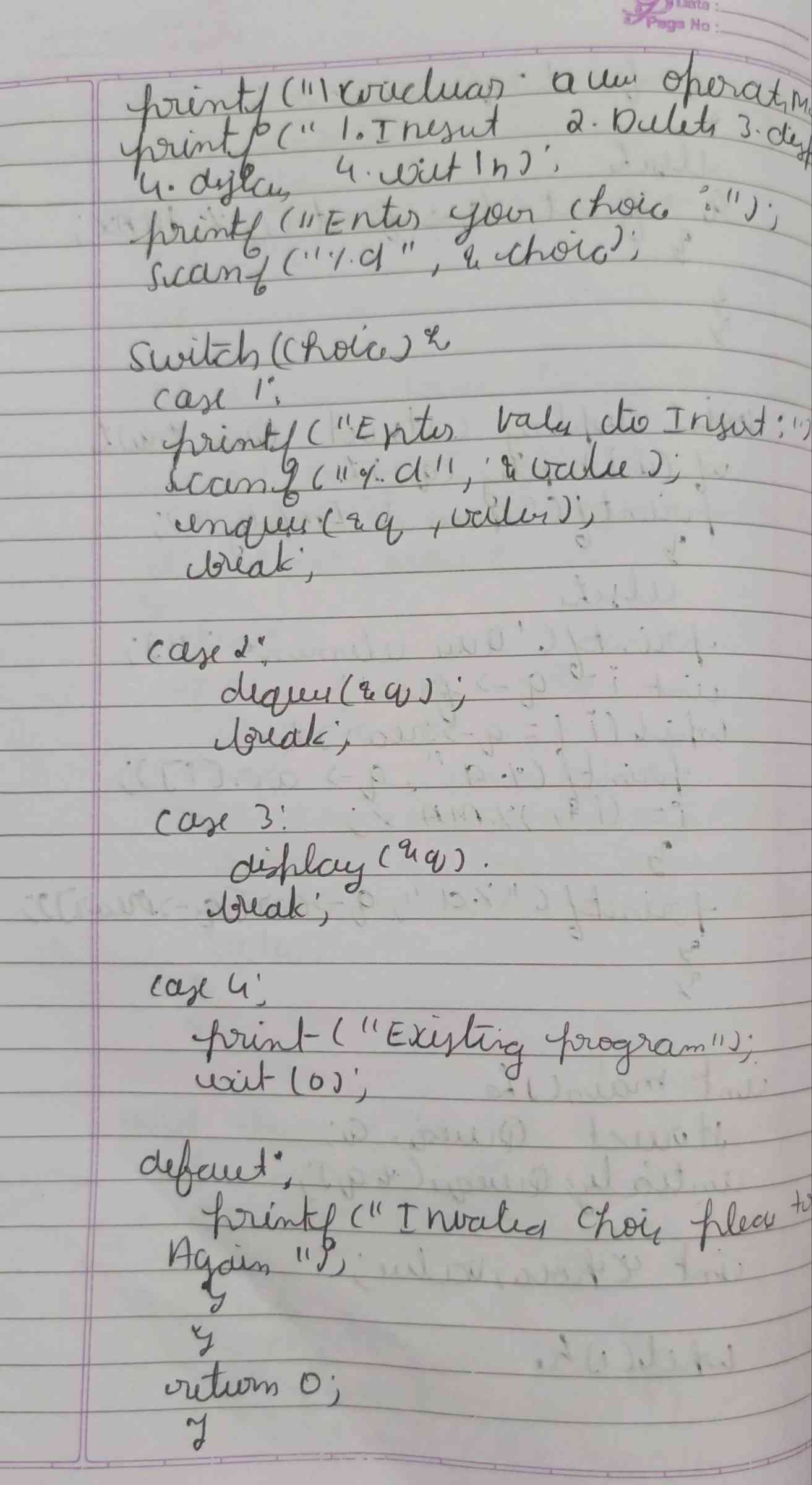
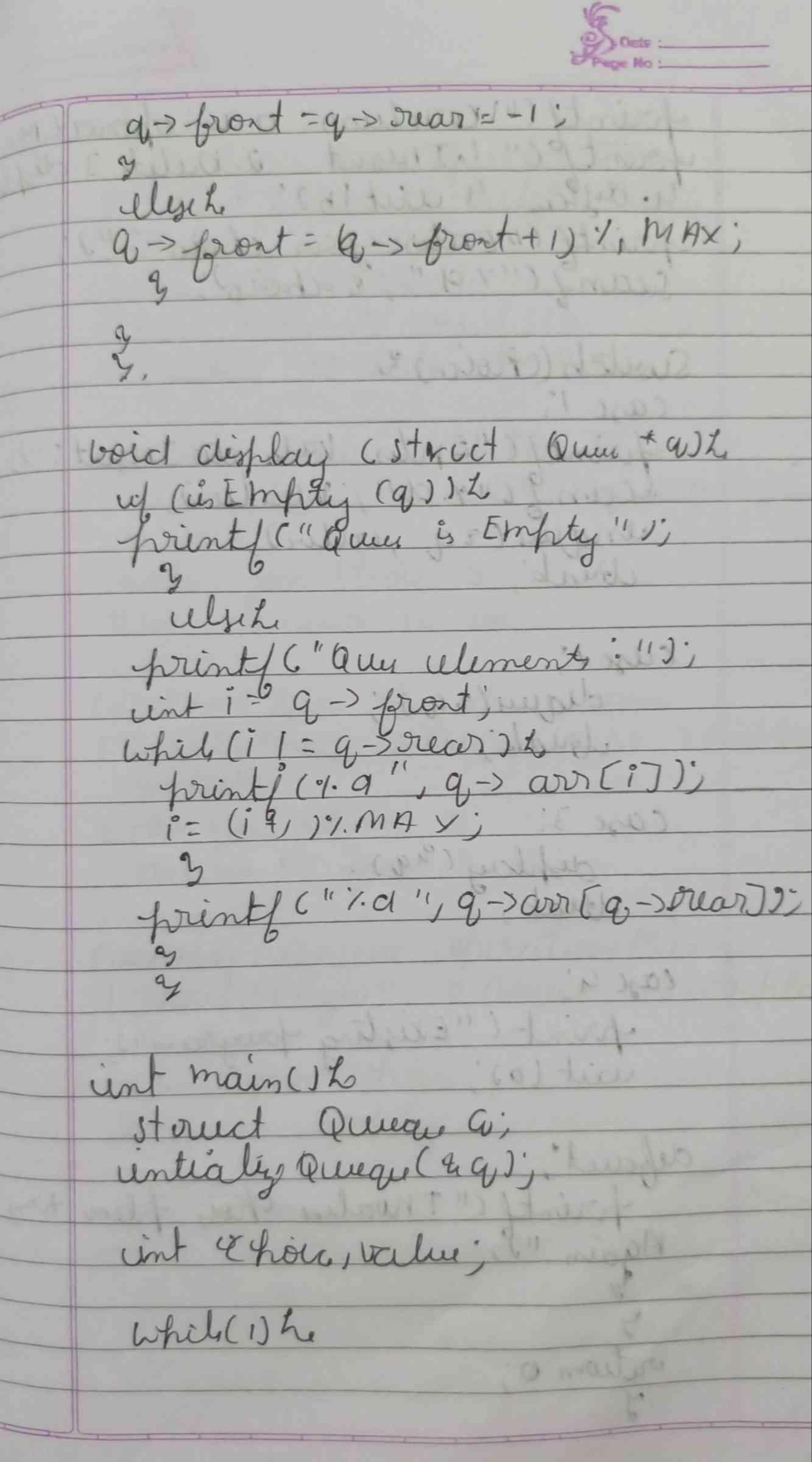
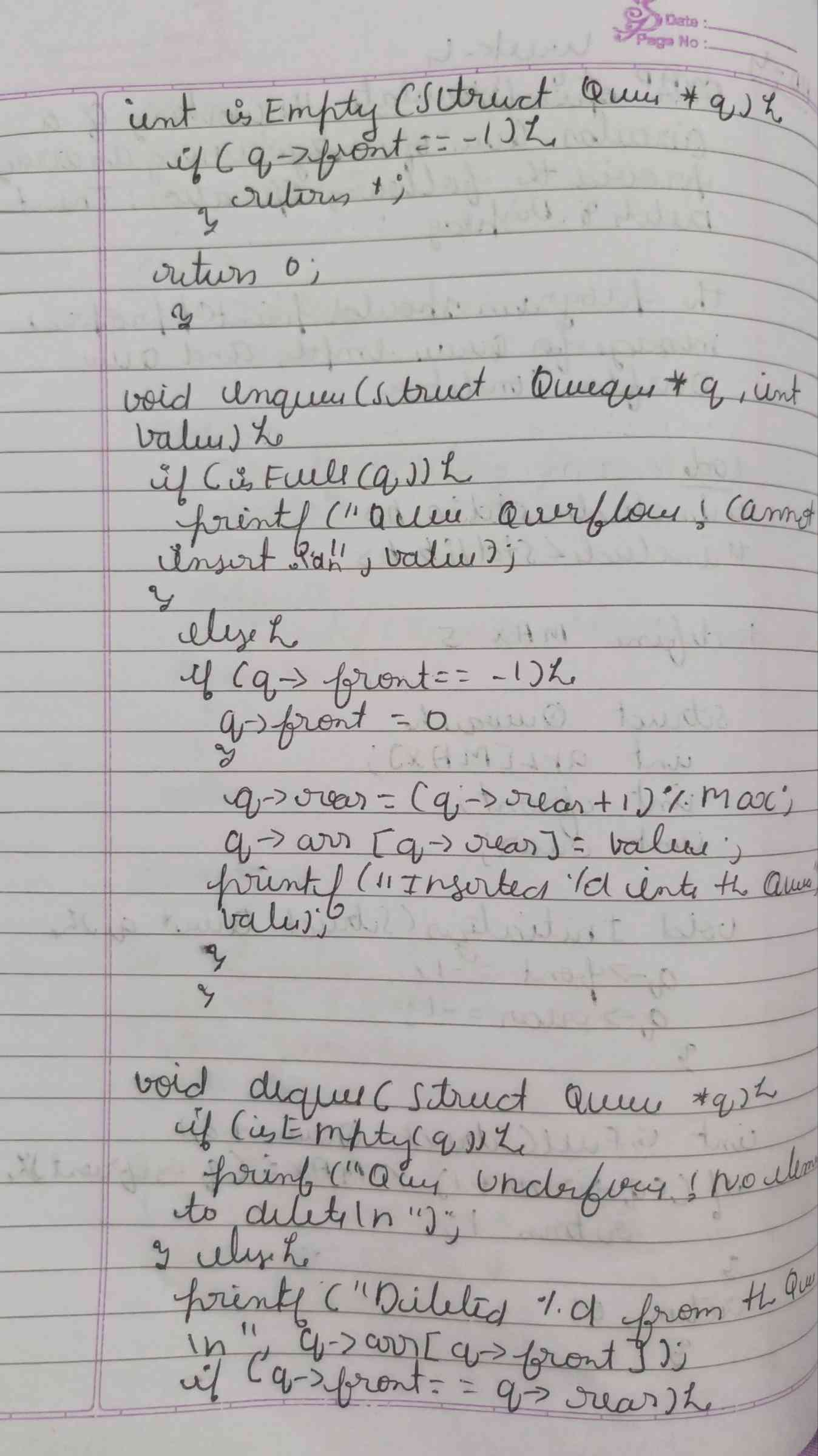
**LABORATORY PROGRAM – 4**

WAP to simulate the working of a circular queue of integers using an array. Provide the following operations: Insert, Delete & Display The program should print appropriate messagesfor queue empty and queue overflow conditions

**OBSERVATION :**

****

****

**CODE :**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define MAX 5**

**int isFull(int front, int rear) {**

**if ((rear + 1) % MAX == front) {**

**return 1;  }**

**return 0; }**

**int isEmpty(int front, int rear) {**

**if (front == -1) {**

**return 1;**

**}**

**return 0;**

**}**

**void insert(int queue[], int \*front, int \*rear, int value) {**

**if (isFull(\*front, \*rear)) {**

**printf("Queue Overflow! Cannot insert %d\n", value);**

**return;**

**}**

**if (\*front == -1) {**

**\*front = \*rear = 0;**

**} else {**

**\*rear = (\*rear + 1) % MAX;**

**}**

**queue[\*rear] = value;**

**printf("%d inserted into the queue\n", value);**

**}**

**void delete(int queue[], int \*front, int \*rear) {**

**if (isEmpty(\*front, \*rear)) {**

**printf("Queue Underflow! No element to delete\n");**

**return;**

**}**

**int deletedValue = queue[\*front];**

**printf("%d deleted from the queue\n", deletedValue);**

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

**\*front = \*rear = -1;**

**} else {**

**\*front = (\*front + 1) % MAX;**

**}**

**}**

**void display(int queue[], int front, int rear) {**

**if (isEmpty(front, rear)) {**

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

**return;**

**}**

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

**if (front <= rear) {**

**for (int i = front; i <= rear; i++) {**

**printf("%d ", queue[i]);**

**}**

**} else {**

**for (int i = front; i < MAX; i++) {**

**printf("%d ", queue[i]);**

**}**

**for (int i = 0; i <= rear; i++) {**

**printf("%d ", queue[i]);**

**}**

**}**

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

**}**

**int main() {**

**int queue[MAX];**

**int front = -1, rear = -1;**

**int choice, value;**

**while (1) {**

**printf("\nCircular Queue Operations:\n");**

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

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

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

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

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

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

**switch (choice) {**

**case 1:**

**printf("Enter the value to insert: ");**

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

**insert(queue, &front, &rear, value);**

**break;**

**case 2:**

**delete(queue, &front, &rear);**

**break;**

**case 3:**

**display(queue, front, rear);**

**break;**

**case 4:**

**exit(0);**

**default:**

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

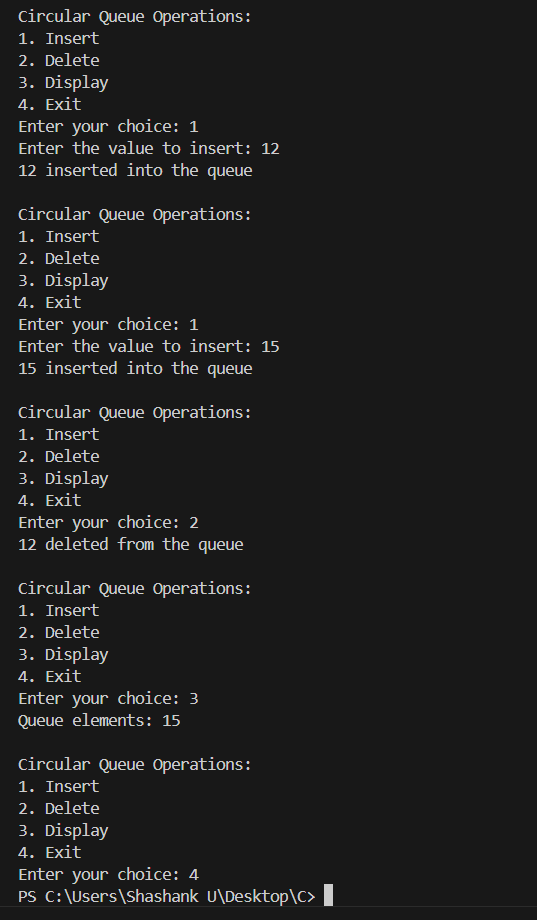
**}**

**}**

**return 0;**

**}**

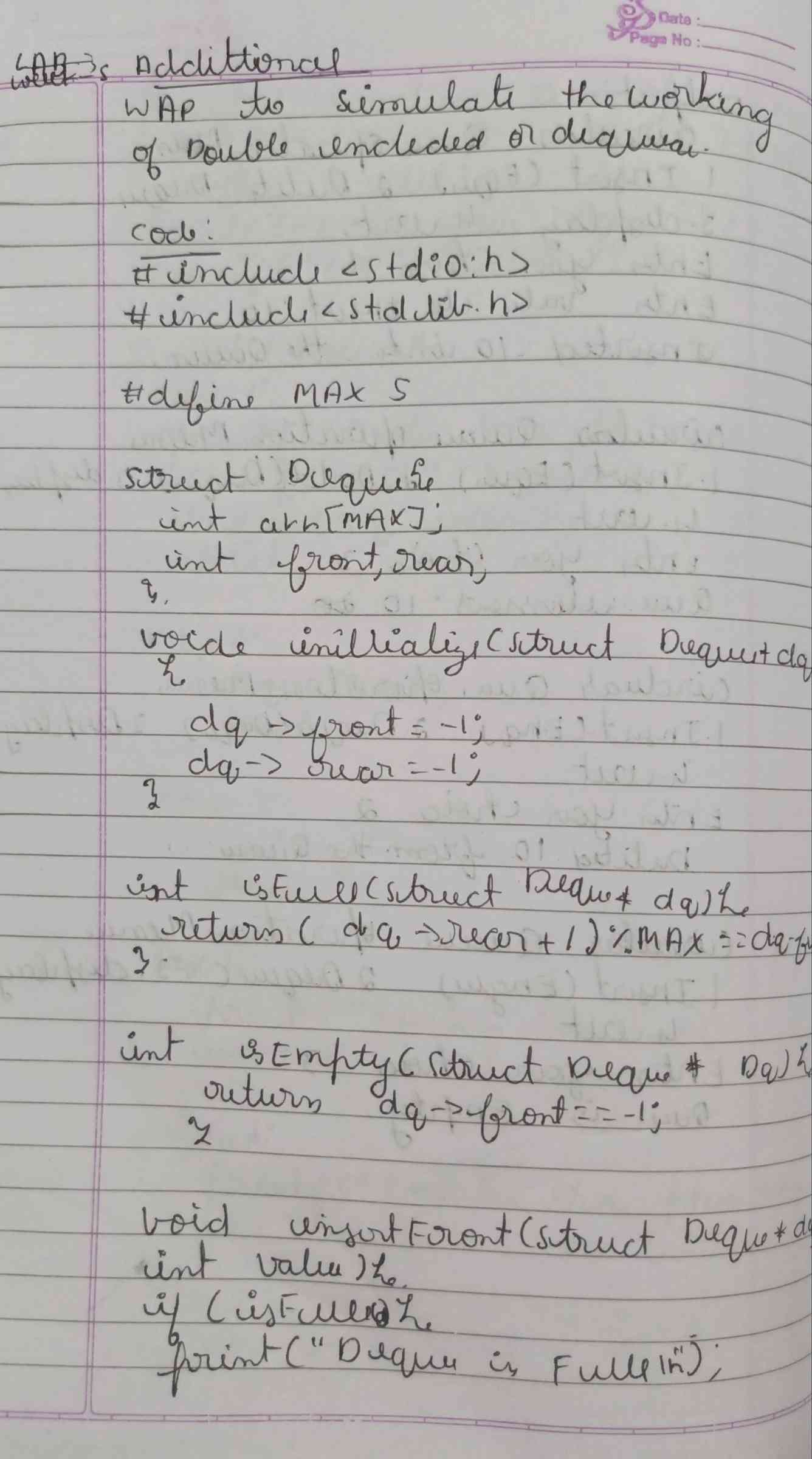
**OUTPUT :**

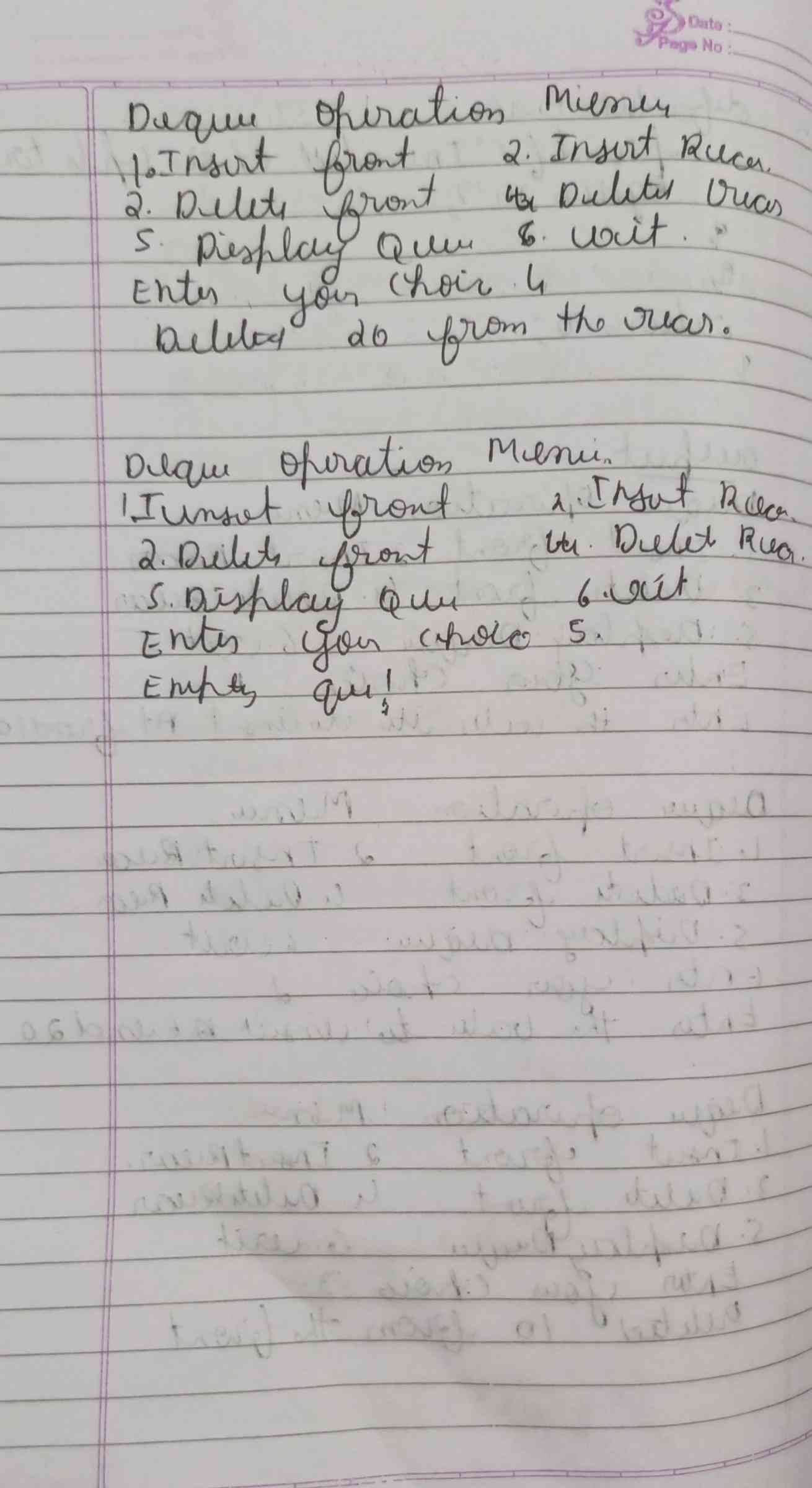
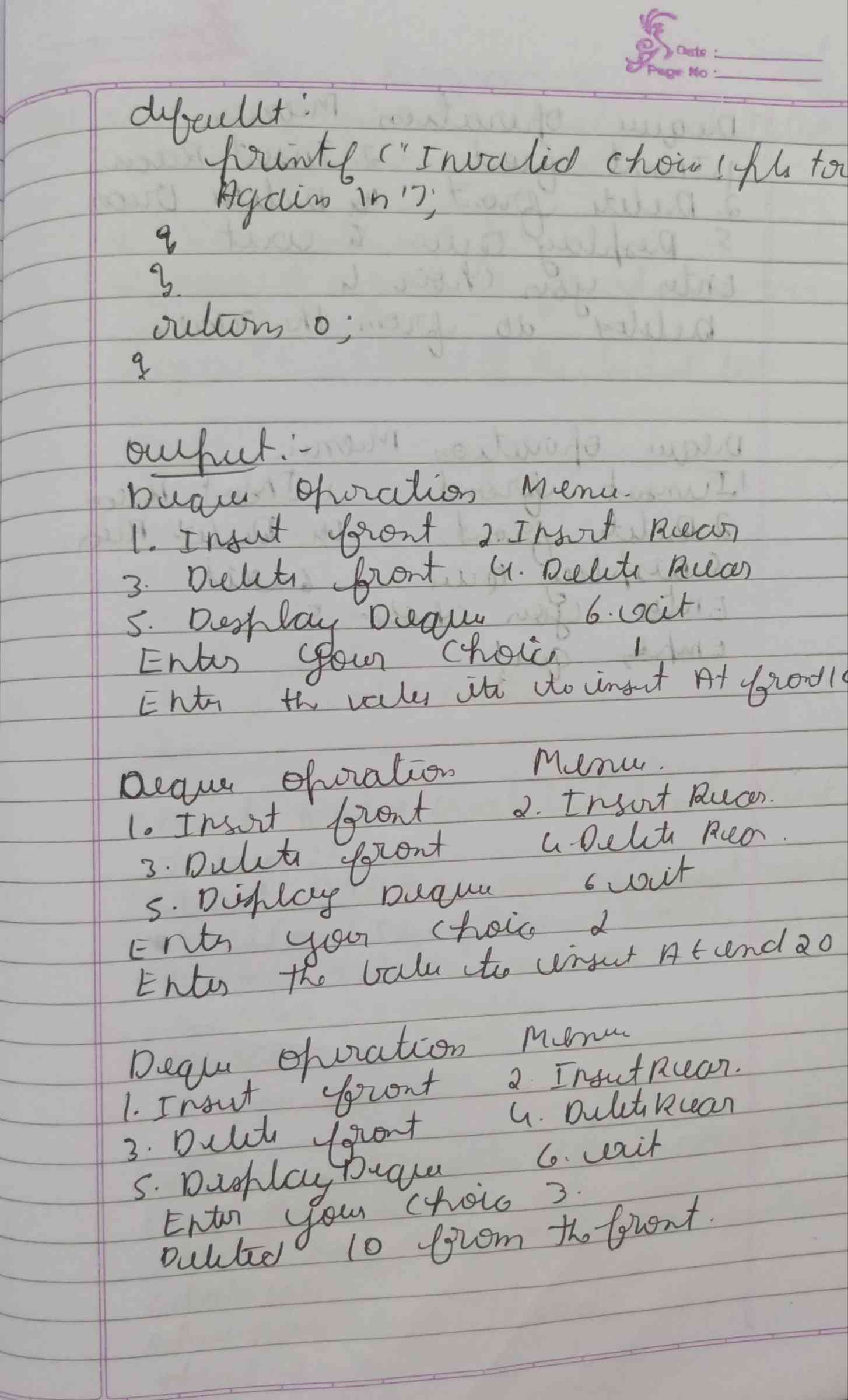
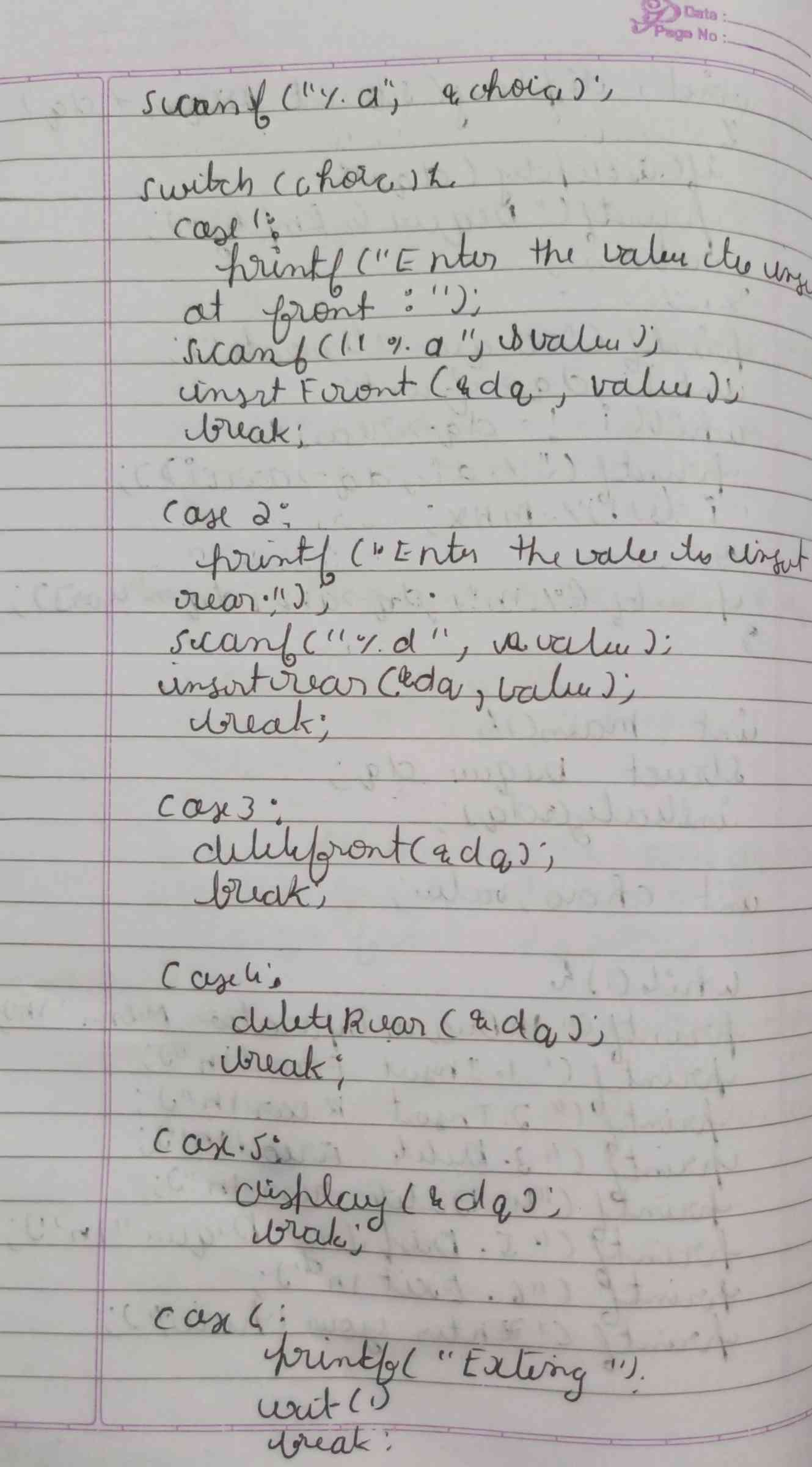
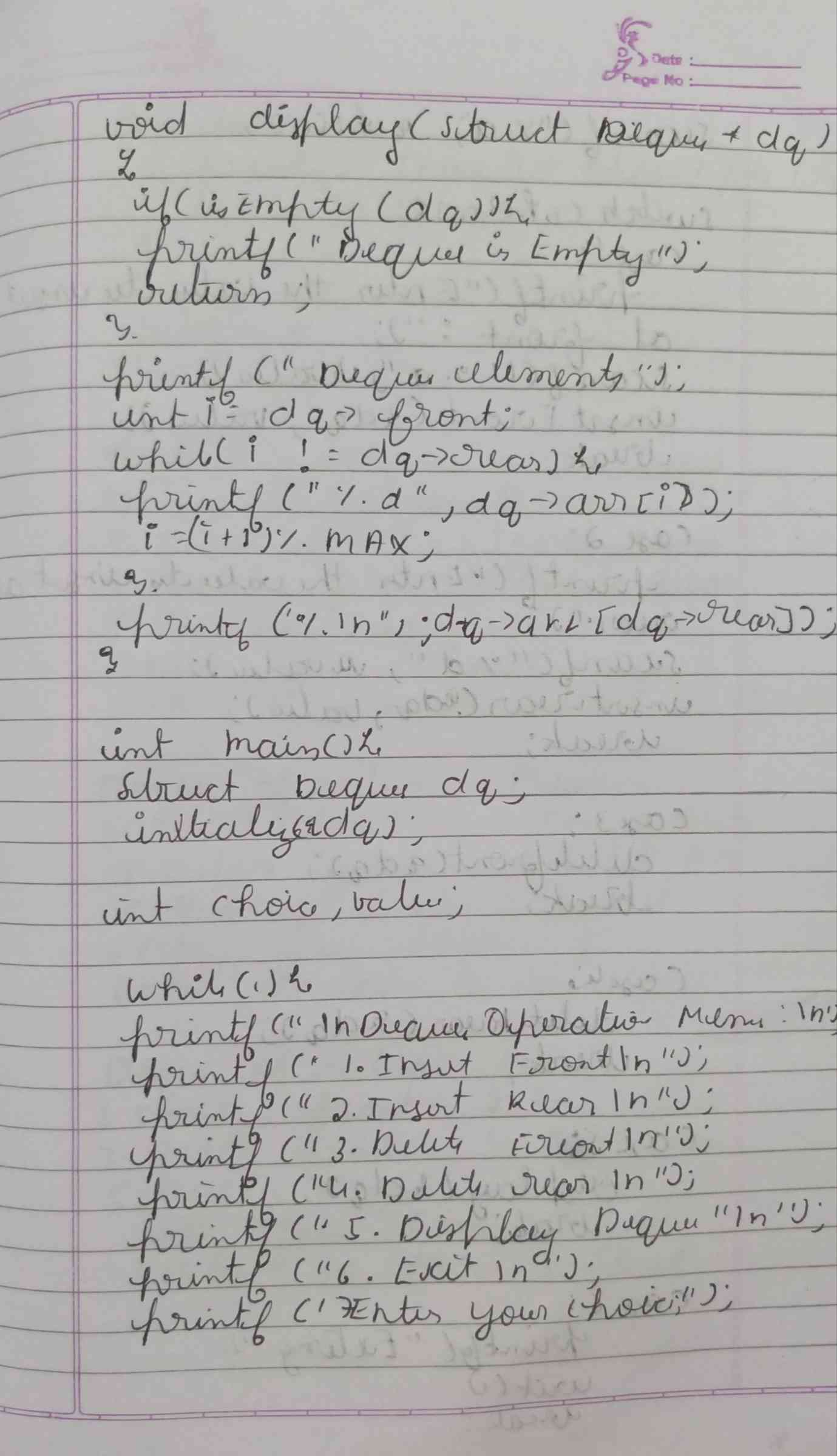
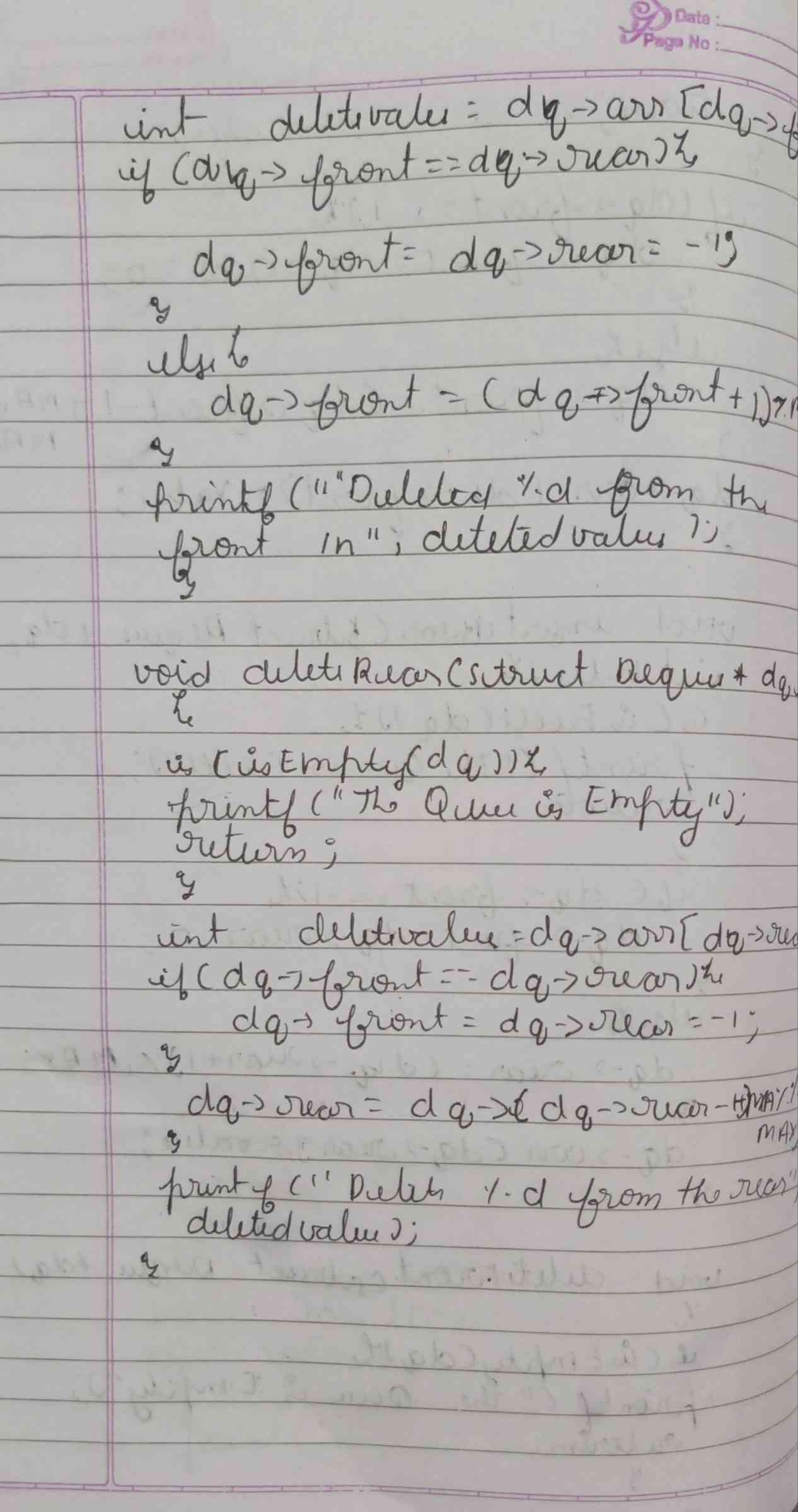
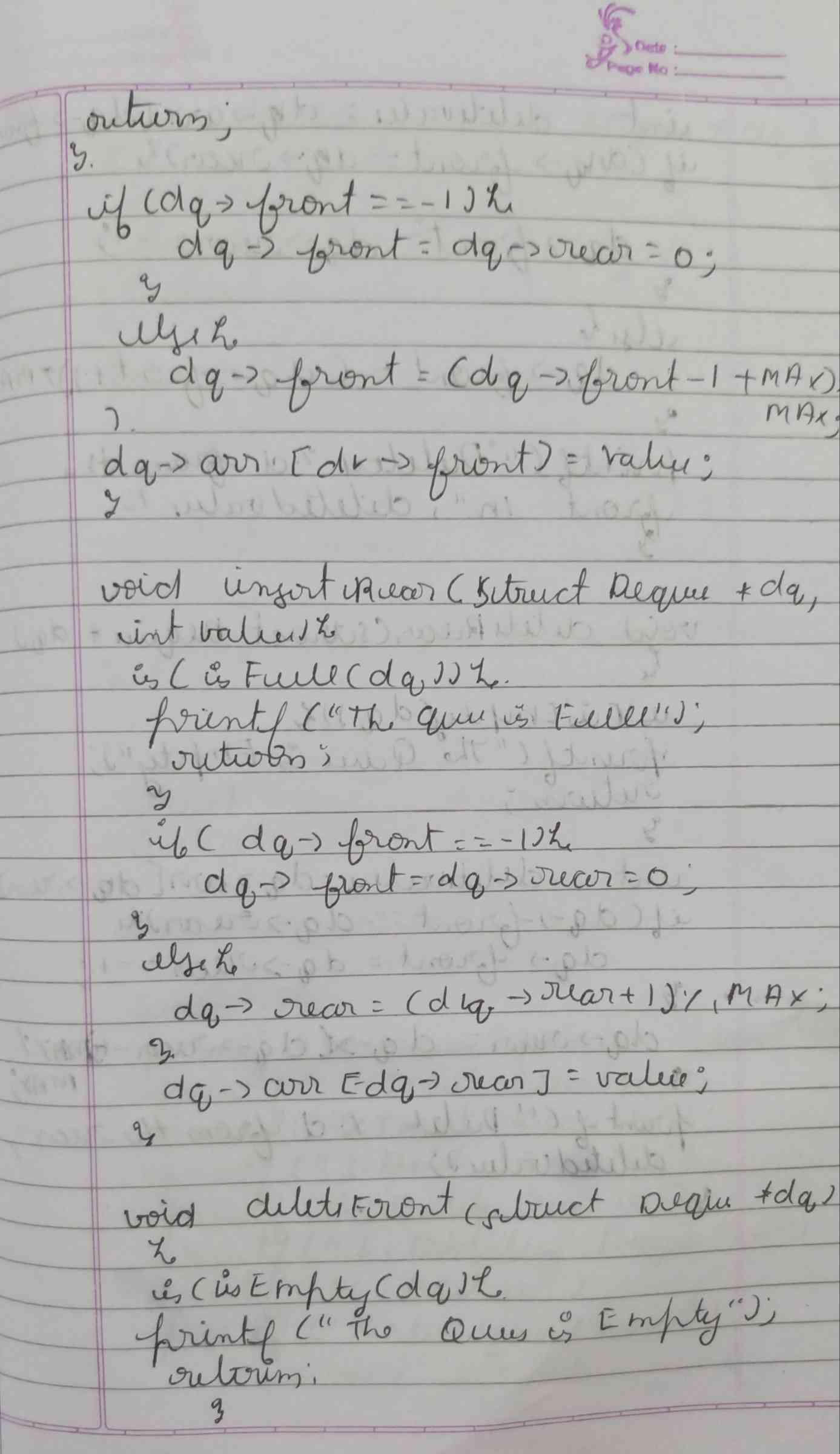


**ADDITTIONAL :**

WAP to simulate the working of Double ended queue or deque.

**OBSERVATION :**





**CODE :**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define MAX 5**

**struct Deque {**

**int arr[MAX];**

**int front, rear;**

**};**

**void initialize(struct Deque\* dq) {**

**dq->front = -1;**

**dq->rear = -1;**

**}**

**int isFull(struct Deque\* dq) {**

**return (dq->rear + 1) % MAX == dq->front;**

**}**

**int isEmpty(struct Deque\* dq) {**

**return dq->front == -1;**

**}**

**void insertFront(struct Deque\* dq, int value) {**

**if (isFull(dq)) {**

**printf("Que is full\n");**

**return;**

**}**

**if (dq->front == -1) {**

**dq->front = dq->rear = 0;**

**} else {**

**dq->front = (dq->front - 1 + MAX) % MAX;**

**}**

**dq->arr[dq->front] = value;**

**}**

**void insertRear(struct Deque\* dq, int value) {**

**if (isFull(dq)) {**

**printf("Que is full.\n");**

**return;**

**}**

**if (dq->front == -1) {**

**dq->front = dq->rear = 0;**

**} else {**

**dq->rear = (dq->rear + 1) % MAX;**

**}**

**dq->arr[dq->rear] = value;**

**}**

**void deleteFront(struct Deque\* dq) {**

**if (isEmpty(dq)) {**

**printf("Que is Empty.\n");**

**return;**

**}**

**int deletedValue = dq->arr[dq->front];**

**if (dq->front == dq->rear) {**

**dq->front = dq->rear = -1;**

**} else {**

**dq->front = (dq->front + 1) % MAX;**

**}**

**printf("Deleted %d from the front.\n", deletedValue);**

**}**

**void deleteRear(struct Deque\* dq) {**

**if (isEmpty(dq)) {**

**printf("Que is Empty.\n");**

**return;**

**}**

**int deletedValue = dq->arr[dq->rear];**

**if (dq->front == dq->rear) {**

**dq->front = dq->rear = -1;**

**} else {**

**dq->rear = (dq->rear - 1 + MAX) % MAX;**

**}**

**printf("Deleted %d from the rear.\n", deletedValue);**

**}**

**void display(struct Deque\* dq) {**

**if (isEmpty(dq)) {**

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

**return;**

**}**

**printf("Deque elements: ");**

**int i = dq->front;**

**while (i != dq->rear) {**

**printf("%d ", dq->arr[i]);**

**i = (i + 1) % MAX;**

**}**

**printf("%d\n", dq->arr[dq->rear]);**

**}**

**int main() {**

**struct Deque dq;**

**initialize(&dq);**

**int choice, value;**

**while(1) {**

**printf("\nDeque Operations Menu:\n");**

**printf("1. Insert Front ");**

**printf("2. Insert Rear ");**

**printf("3. Delete Front ");**

**printf("4. Delete Rear ");**

**printf("5. Display Deque ");**

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

**printf("\nEnter your choice: ");**

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

**switch (choice) {**

**case 1:**

**printf("Enter value to insert at front: ");**

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

**insertFront(&dq, value);**

**break;**

**case 2:**

**printf("Enter value to insert at rear: ");**

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

**insertRear(&dq, value);**

**break;**

**case 3:**

**deleteFront(&dq);**

**break;**

**case 4:**

**deleteRear(&dq);**

**break;**

**case 5:**

**display(&dq);**

**break;**

**case 6:**

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

**exit(1);**

**break;**

**default:**

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

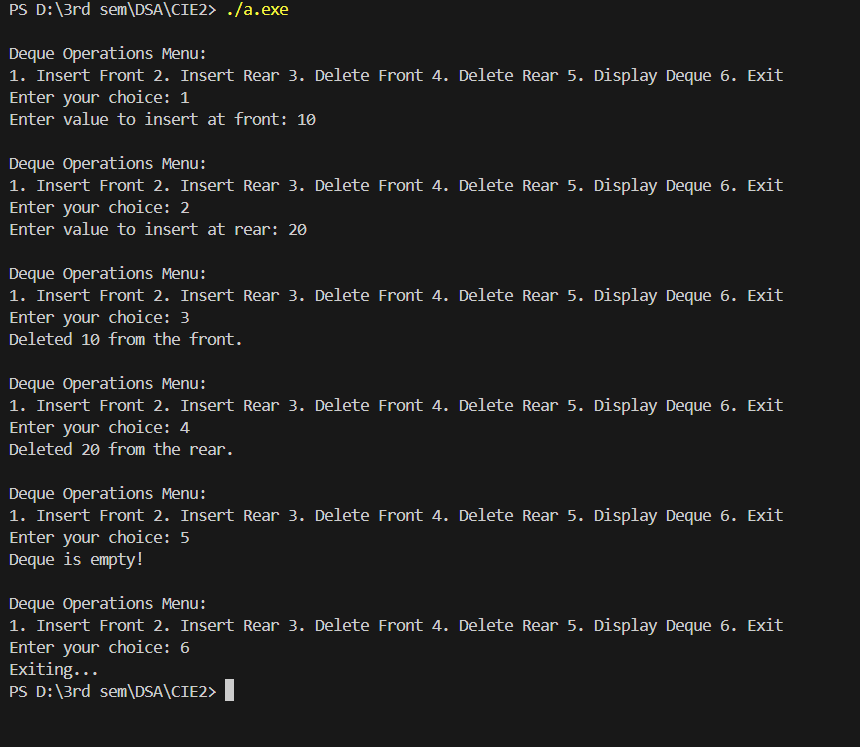
**}**

**}**

**return 0;**

**}**

**OUTPUT :**

****