//LINEAR QUEUE IMPLEMENTATION IN C//

#include<stdio.h>

#include<stdlib.h>

#define SIZE 6

void enqueue();

void dequeue();

void display();

int queue[SIZE], front = -1, rear = -1;

int choice,value;

int main()

{

printf("1.Insertion\n 2.Deletion\n 3.Display\n 4.search");

do

{

printf("\nEnter your choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:

{

printf("Enter the element you need to insert:");

scanf("%d",&value);

enqueue();

break;

}

case 2:

{

dequeue();

break;

}

case 3:

{

display();

break;

}

default:

{

printf("Wrong selection");

}

}

}while(choice!=3);

}

void enqueue()

{

if (rear == SIZE - 1)

printf("\nQueue is Full!!");

else

{

if(front == -1)

front = 0;

rear++;

queue[rear] = value;

printf("\nInserted -> %d", value);

}

}

void dequeue()

{

if (front == -1)

printf("\nQueue is Empty!");

else if (front > rear)

front = rear = -1;

else

{

printf("\nDeleted : %d", queue[front]);

front++;

}

}

void display()

{

if (rear == -1)

printf("\nQueue is Empty!");

else

{

int i;

printf("\nQueue elements are:\n");

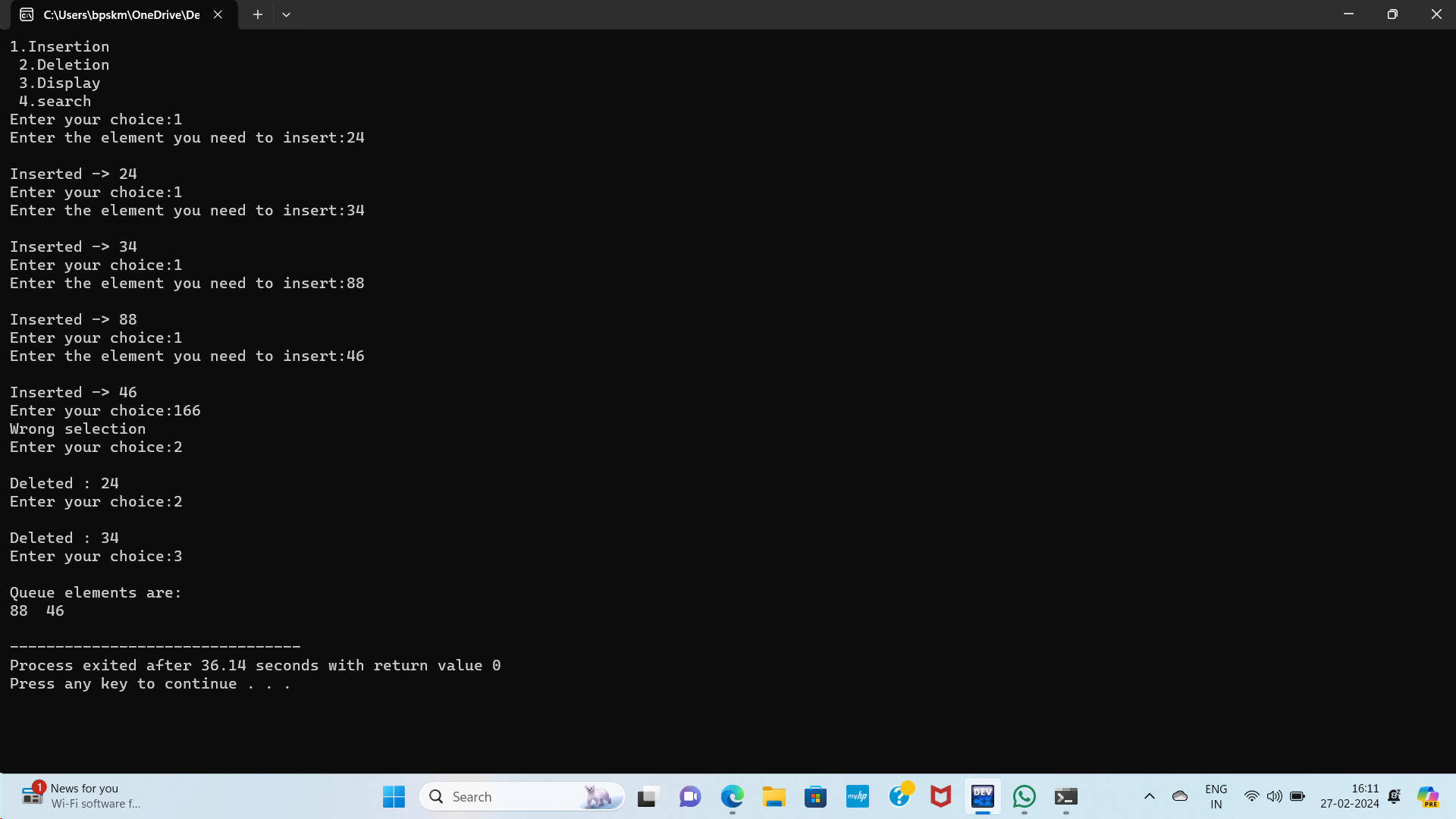
for (i = front; i <= rear; i++)

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

}

printf("\n");

}



//CIRCULAR QUEUE IMPLEMENTATION IN C//

#include<stdio.h>

#include<stdlib.h>

#define SIZE 10

void enqueue();

void dequeue();

void display();

int queue[SIZE],front=-1,rear=-1;

int value,choice;

int main()

{

printf("1.Insertion\n 2.Deletion\n 3.Display\n");

do

{

printf("\nEnter your choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:

{

printf("Enter the value to be inserted:");

scanf("%d",&value);

enqueue();

break;

}

case 2:

{

dequeue();

break;

}

case 3:

{

display();

break;

}

default:

{

printf("Wrong selection");

}

}

}while(choice!=3);

}

void enqueue()

{

if ((front==0 && rear==SIZE-1)||front==rear+1)

printf("\nQueue is Empty!");

else

{

if(front == -1)

front=0;

rear = (rear+1)%SIZE;

queue[rear] = value;

}

}

void dequeue()

{

if(front == -1)

printf("\nQueue is empty");

else

{

printf("Deleted element:%d",queue[front]);

front = (front+1)%SIZE;

if(front == rear)

front = rear = -1;

}

}

void display()

{

if (front == -1)

printf("\nQueue is Empty!");

else

{

int i;

printf("\nQueue elements are:\n");

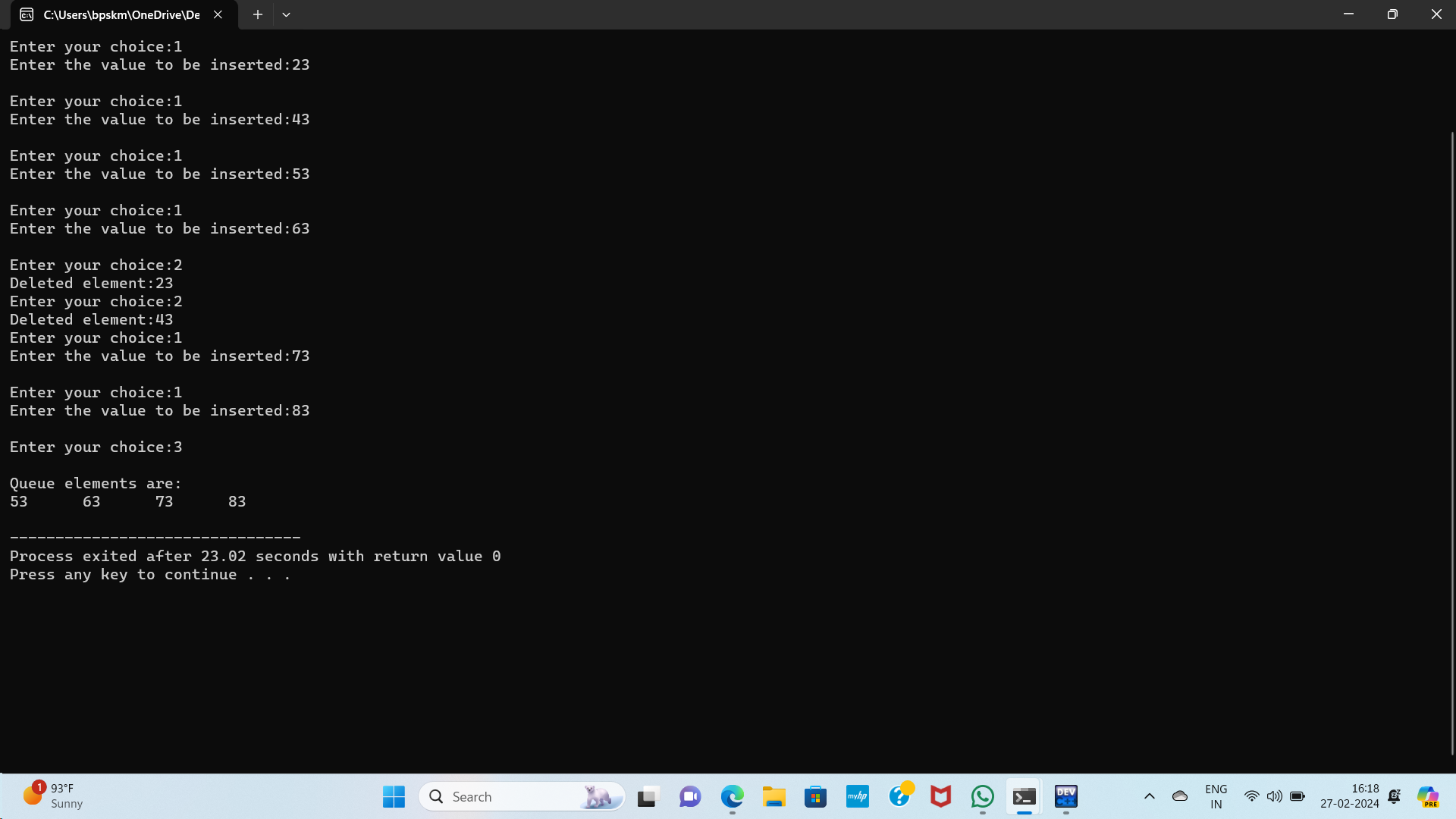
for (i = front; i <= rear; i=(i+1)%SIZE)

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

}

printf("\n");

}



// C program to solve N Queen Problem using backtracking

#define N 4

#include<stdbool.h>

#include<stdio.h>

void printSolution(int board[N][N])

{

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

if(board[i][j])

printf("Q ");

else

printf(". ");

}

printf("\n");

}

}

bool isSafe(int board[N][N], int row, int col)

{

int i, j;

for (i = 0; i < col; i++)

if (board[row][i])

return false;

for (i = row, j = col; i >= 0 && j >= 0; i--, j--)

if (board[i][j])

return false;

for (i = row, j = col; j >= 0 && i < N; i++, j--)

if (board[i][j])

return false;

return true;

}

bool solveNQUtil(int board[N][N], int col)

{

if (col >= N)

return true;

for (int i = 0; i < N; i++)

{

if (isSafe(board, i, col))

{

board[i][col] = 1;

if (solveNQUtil(board, col + 1))

return true;

board[i][col] = 0;

}

}

return false;

}

bool solveNQ()

{

int board[N][N] = { { 0, 0, 0, 0 },

{ 0, 0, 0, 0 },

{ 0, 0, 0, 0 },

{ 0, 0, 0, 0 } };

if (solveNQUtil(board, 0) == false)

{

printf("Solution does not exist");

return false;

}

printSolution(board);

return true;

}

int main()

{

solveNQ();

return 0;

}

