**DATA STRUCTURES LAB**

**LAB TASK – 5**

**PROGRAM – 1**

**#include <stdio.h>**

**#define max 10**

**int queue[max];**

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

**int main()**

**{**

**int choice,val;**

**while (1) {**

**printf("\nAVAILABLE CHOICES ARE: \n");**

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

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

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

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

**printf("5. Exit\n");**

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

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

**switch (choice)**

**{**

**case 1:**

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

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

**enqueue(val);**

**break;**

**case 2:**

**dequeue();**

**break;**

**case 3:**

**printf("Enter key value: ");**

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

**find(val);**

**break;**

**case 4:**

**display();**

**break;**

**case 5:**

**exit(0);**

**default:**

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

**}**

**}**

**return 0;**

**}**

**void enqueue(int val) {**

**if(rear == max-1)**

**{**

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

**}**

**else**

**{**

**rear++;**

**queue[rear] = val;**

**printf("enqueue operation is successful\n", val);**

**}**

**}**

**void dequeue()**

**{**

**if (front == -1&& rear==-1)**

**{**

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

**}**

**else**

**{**

**printf("%d deleted from the queue\ndequeue operation is successful\n",queue[front]);**

**front++;**

**if(front>rear)**

**{**

**front=rear=-1;**

**}**

**}**

**}**

**void find(int key)**

**{**

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

**{**

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

**}**

**else**

**{**

**int found = 0;**

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

**{**

**if (queue[i] == key)**

**{**

**printf("Element found at position %d\n", i - front + 1);**

**found = 1;**

**break;**

**}**

**}**

**if (!found) {**

**printf("Element not found in the queue\n");**

**}**

**}**

**}**

**void display()**

**{**

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

**{**

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

**}**

**else**

**{**

**printf("Queue elements are:\n");**

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

**{**

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

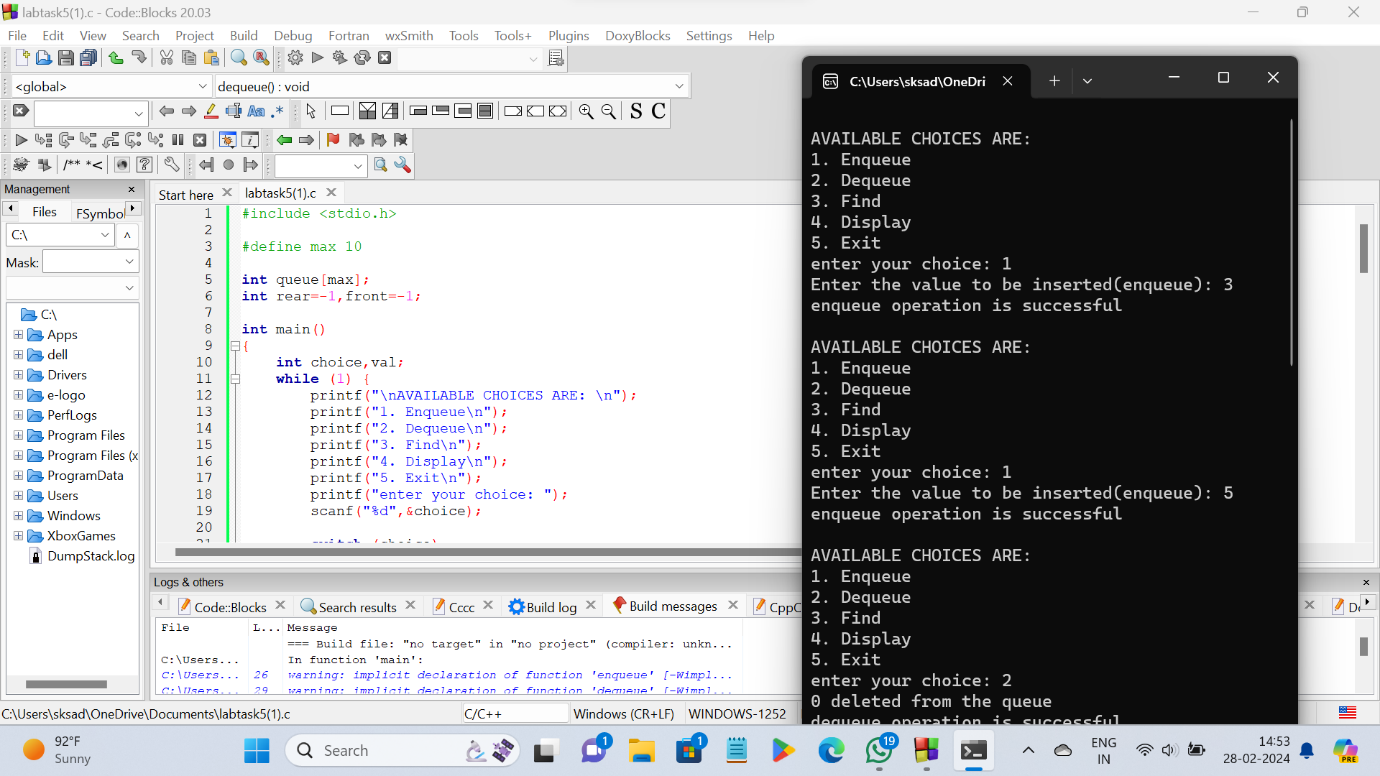
**}**

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

**}**

**}**

**OUTPUT :**

****

**PROGRAM – 2**

**#include <stdio.h>**

**#define max 10**

**int queue[max];**

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

**int main() {**

**printf("IMPLEMENTATION OF CIRCULAR QUEUE USING ARRAYS:");**

**int choice, val;**

**while (1) {**

**printf("\nAVAILABLE CHOICES ARE: \n");**

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

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

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

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

**printf("5. Exit\n");**

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

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

**switch (choice) {**

**case 1:**

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

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

**enqueue(val);**

**break;**

**case 2:**

**dequeue();**

**break;**

**case 3:**

**printf("Enter key value: ");**

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

**find(val);**

**break;**

**case 4:**

**display();**

**break;**

**case 5:**

**return 0;**

**default:**

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

**}**

**}**

**return 0;**

**}**

**void enqueue(int val)**

**{**

**if ((front == 0 && rear == max - 1) || (rear == (front - 1) % (max - 1)))**

**{**

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

**}**

**else if (front == -1 && rear == -1) {**

**front = rear = 0;**

**queue[rear] = val;**

**printf("enqueue operation is successful\n");**

**} else if (rear == max - 1 && front != 0) {**

**rear = 0;**

**queue[rear] = val;**

**printf("enqueue operation is successful\n");**

**} else {**

**rear++;**

**queue[rear] = val;**

**printf("enqueue operation is successful\n");**

**}**

**}**

**void dequeue()**

**{**

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

**{**

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

**}**

**else if (front == rear)**

**{**

**printf("%d deleted from the queue\ndequeue operation is successful\n", queue[front]);**

**front = rear = -1;**

**}**

**else if (front == max - 1)**

**{**

**printf("%d deleted from the queue\ndequeue operation is successful\n", queue[front]);**

**front = 0;**

**}**

**else**

**{**

**printf("%d deleted from the queue\ndequeue operation is successful\n", queue[front]);**

**front++;**

**}**

**}**

**void find(int key)**

**{**

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

**{**

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

**return;**

**}**

**int i = front;**

**int position = 1;**

**do {**

**if (queue[i] == key) {**

**printf("Element found at position %d\n", position);**

**return;**

**}**

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

**position++;**

**} while (i != (rear + 1) % max);**

**printf("Element not found in the queue\n");**

**}**

**void display() {**

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

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

**return;**

**}**

**int i = front;**

**printf("Queue elements are:\n");**

**do {**

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

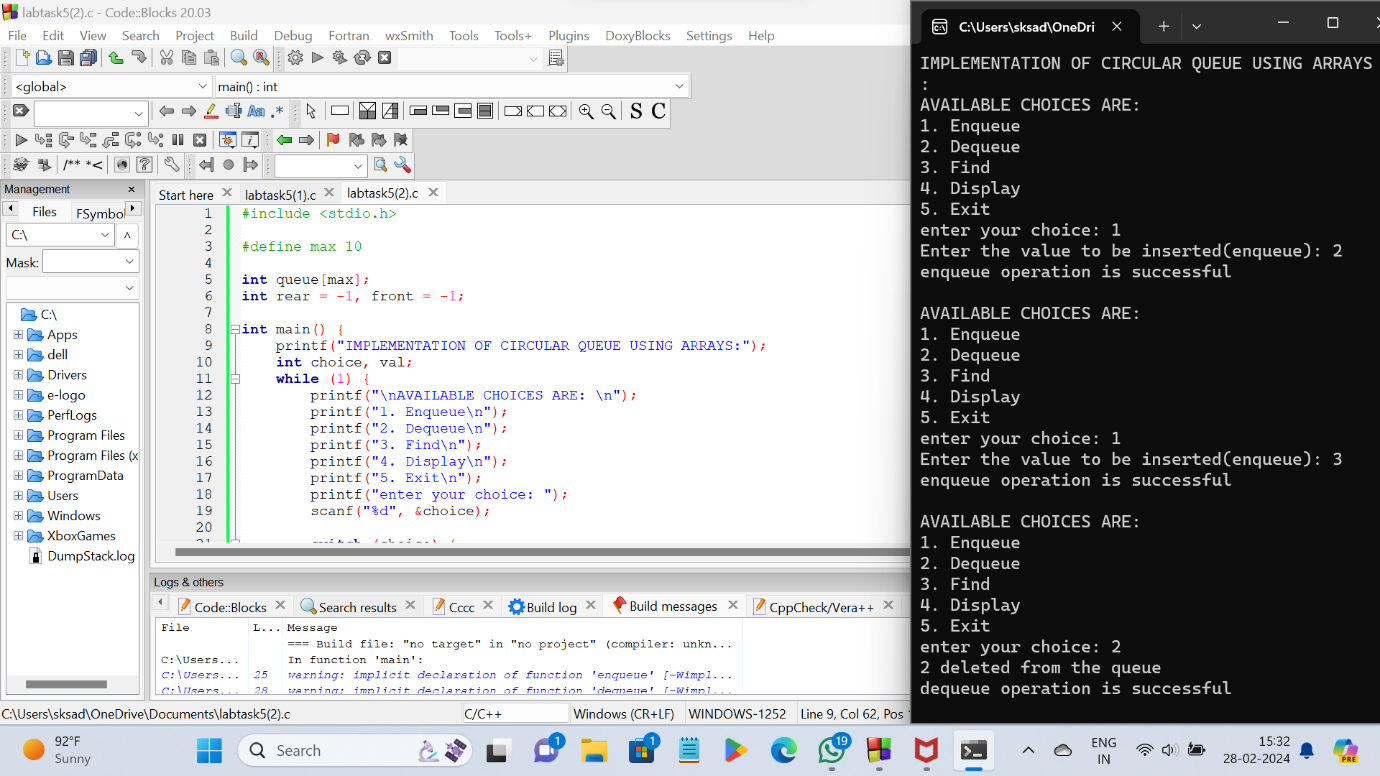
**i = (i + 1) % max;**

**} while (i != (rear + 1) % max);**

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

**}**

**OUTPUT :**

****

**PROGRAM - 3**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define N 4**

**typedef struct {**

**int row;**

**int col;**

**} Queen;**

**Queen stack[N];**

**int top = -1;**

**void push(int row, int col) {**

**if (top == N - 1) {**

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

**exit(1);**

**}**

**top++;**

**stack[top].row = row;**

**stack[top].col = col;**

**}**

**void pop() {**

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

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

**exit(1);**

**}**

**top--;**

**}**

**int isvalid(int row, int col) {**

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

**if (stack[i].col == col || stack[i].row + stack[i].col == row + col || stack[i].row - stack[i].col == row - col) {**

**return 0;**

**}**

**}**

**return 1;**

**}**

**void printSolution() {**

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

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

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

**if (stack[i].col == j) {**

**printf("Q ");**

**} else {**

**printf(". ");**

**}**

**}**

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

**}**

**}**

**void solveNQueens() {**

**int row = 0, col = 0;**

**while (row < N) {**

**while (col < N) {**

**if (isvalid(row, col)) {**

**push(row, col);**

**break;**

**} else {**

**col++;**

**}**

**}**

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

**printf("No solution exists\n");**

**return;**

**}**

**if (col == N) {**

**col = stack[top].col + 1;**

**row = stack[top].row;**

**pop();**

**continue;**

**}**

**if (top == N - 1) {**

**printSolution();**

**col = stack[top].col + 1;**

**row = stack[top].row;**

**pop();**

**continue;**

**}**

**row++;**

**col = 0;**

**}**

**}**

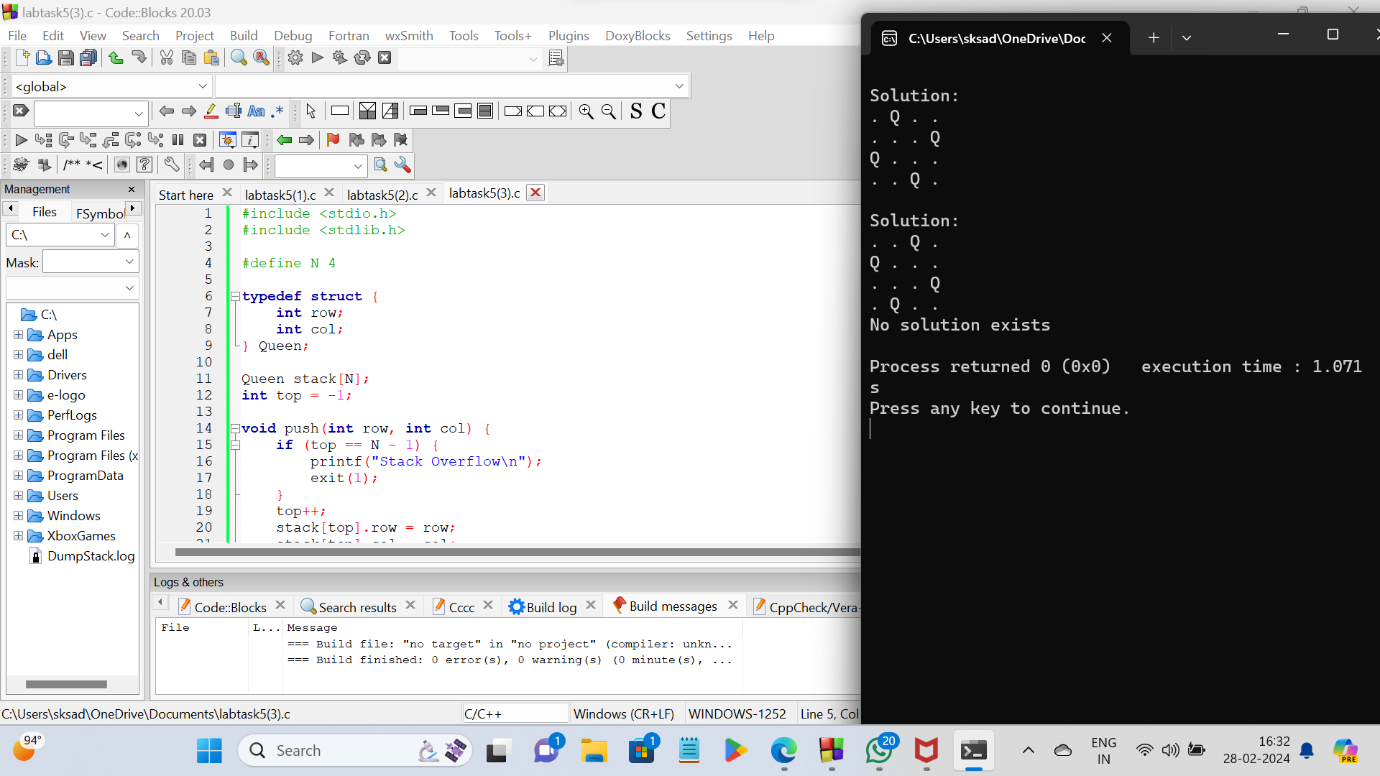
**int main() {**

**solveNQueens();**

**return 0;**

**}**

**OUTPUT :**

****

**\*\*\*\*\*THE END\*\*\*\*\***