Assignment 1

Course: Data Structures

Course id: 14461002

Student id: 202033762

Name: 장민호

Major : 설비소방공학과

Submission Date : 2022_03_06

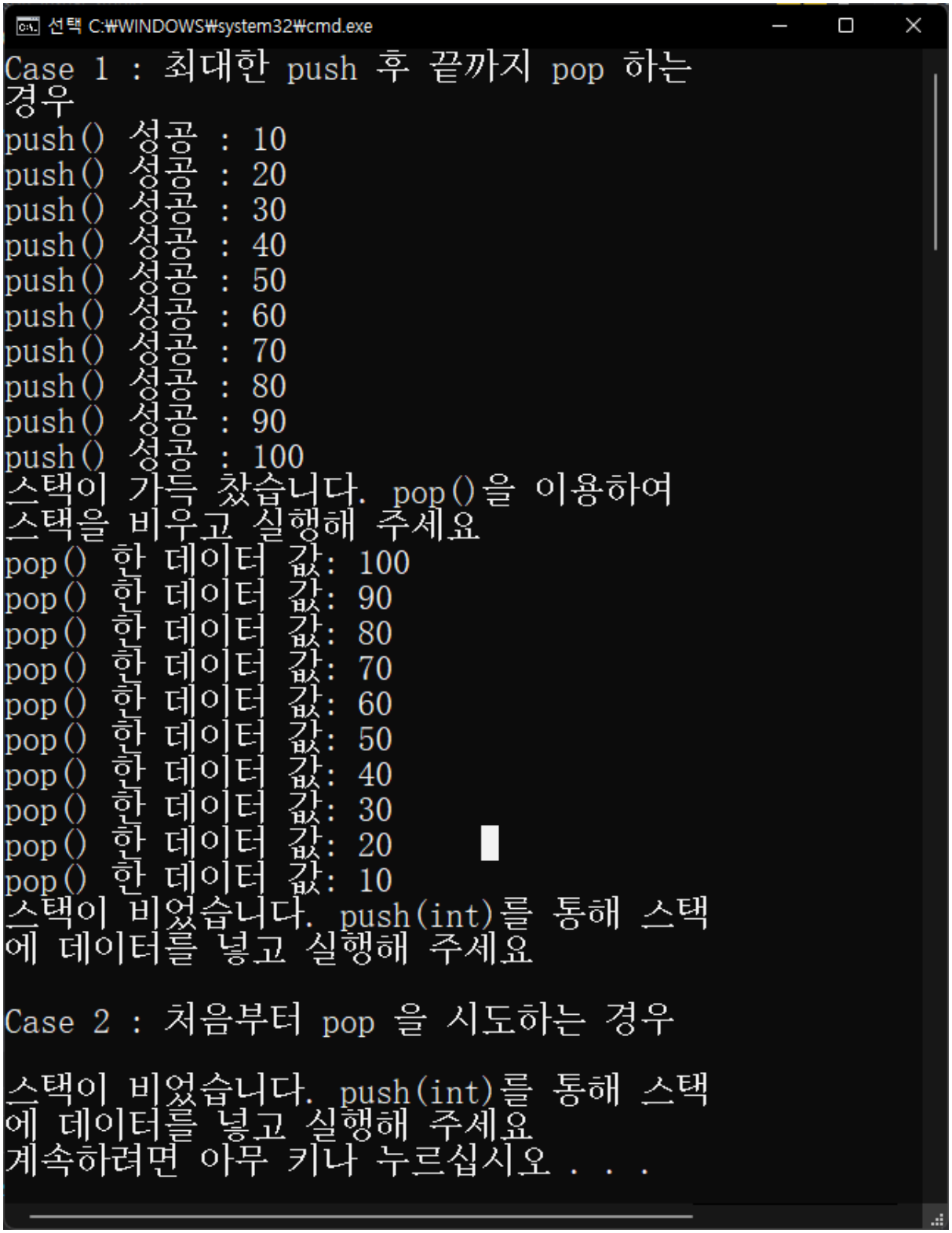
Lab 1-1

Source Code

```c
#include <stdio.h>

/*
0304 Data Structures 과제 LAB 1-1
202033762 장민호

조건
1. 스택 최대크기 10
2. 구현할 function: push(int), pop(), stack_full(), stack_empty()
3. 함수 호출 시 포인터 사용하지 않기
4. scanf, printf 는 반드시 main 에서만 사용
5. push 하기 전 stack_full 검사
6. pop 하기 전 stack_empty 검사
*/

#define TRUE 1
#define FALSE 0
#define MAXSIZE 10

int topIndex = -1; // Stack 의 Top 에 위치한 데이터의 index 값. 초기에는 아무것도 없으므로 -1 이다.
int stack[MAXSIZE];

void push(int num) {
        topIndex++;
        stack[topIndex] = num;
}

int pop() {
        int top = stack[topIndex];
        topIndex--;
        return top;
}

int stack_full() {
        if (topIndex == MAXSIZE-1) {
                return TRUE;
        }
        return FALSE;
}

int stack_empty() {
        if (topIndex < 0) {
                return TRUE;
        }
        return FALSE;
}

int main() {
```

```
	int testNum = 10;

	// Case 1 : 최대한 push 후 끝까지 pop 하는 경우

	printf("Case 1 : 최대한 push 후 끝까지 pop 하는 경우\n");

	while (1) {
		if (stack_full() == TRUE) {
			printf("스택이 가득 찼습니다. pop()을 이용하여 스택을
비우고 실행해 주세요\n");
			break;
		}
		else {
			push(testNum);
			printf("push() 성공 : %d\n", testNum);
			testNum += 10;
		}
	}

	while (1) {
		if (stack_empty() == TRUE) {
			printf("스택이 비었습니다. push(int)를 통해 스택에
데이터를 넣고 실행해 주세요\n");
			break;
		}
		else {
			printf("pop() 한 데이터 값: %d\n",pop());
		}
	}

	// Case 2 : 처음부터 pop 을 시도하는 경우

	printf("\n");
	printf("Case 2 : 처음부터 pop 을 시도하는 경우\n\n");

	if (stack_empty() == TRUE) {
		printf("스택이 비었습니다. push(int)를 통해 스택에 데이터를 넣고
실행해 주세요\n");
	}
	else {
		printf("pop() 한 데이터 값: %d\n", pop());
	}

	return 0;
}
```

Screenshots



LAB 1-2

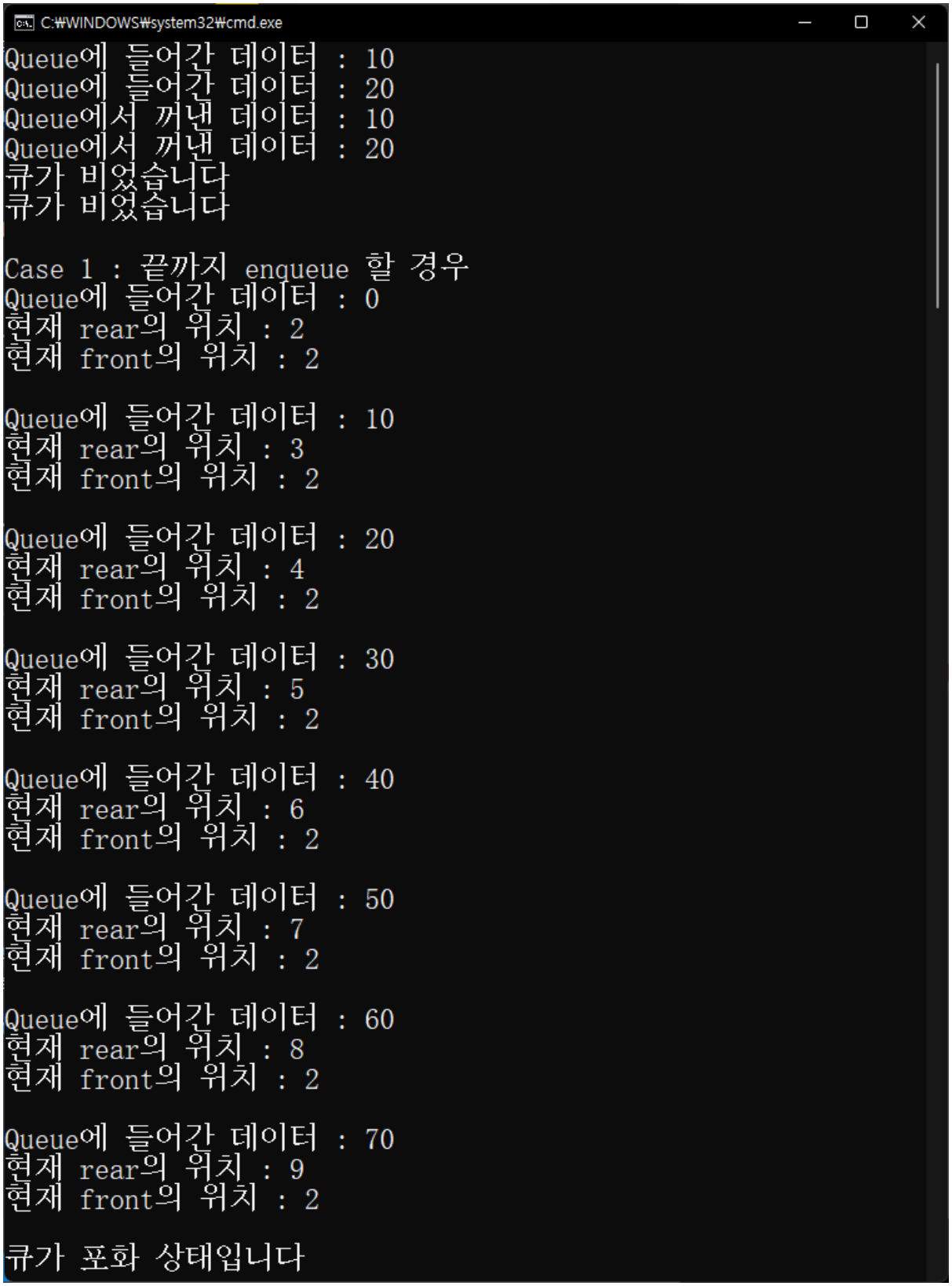
Source Code

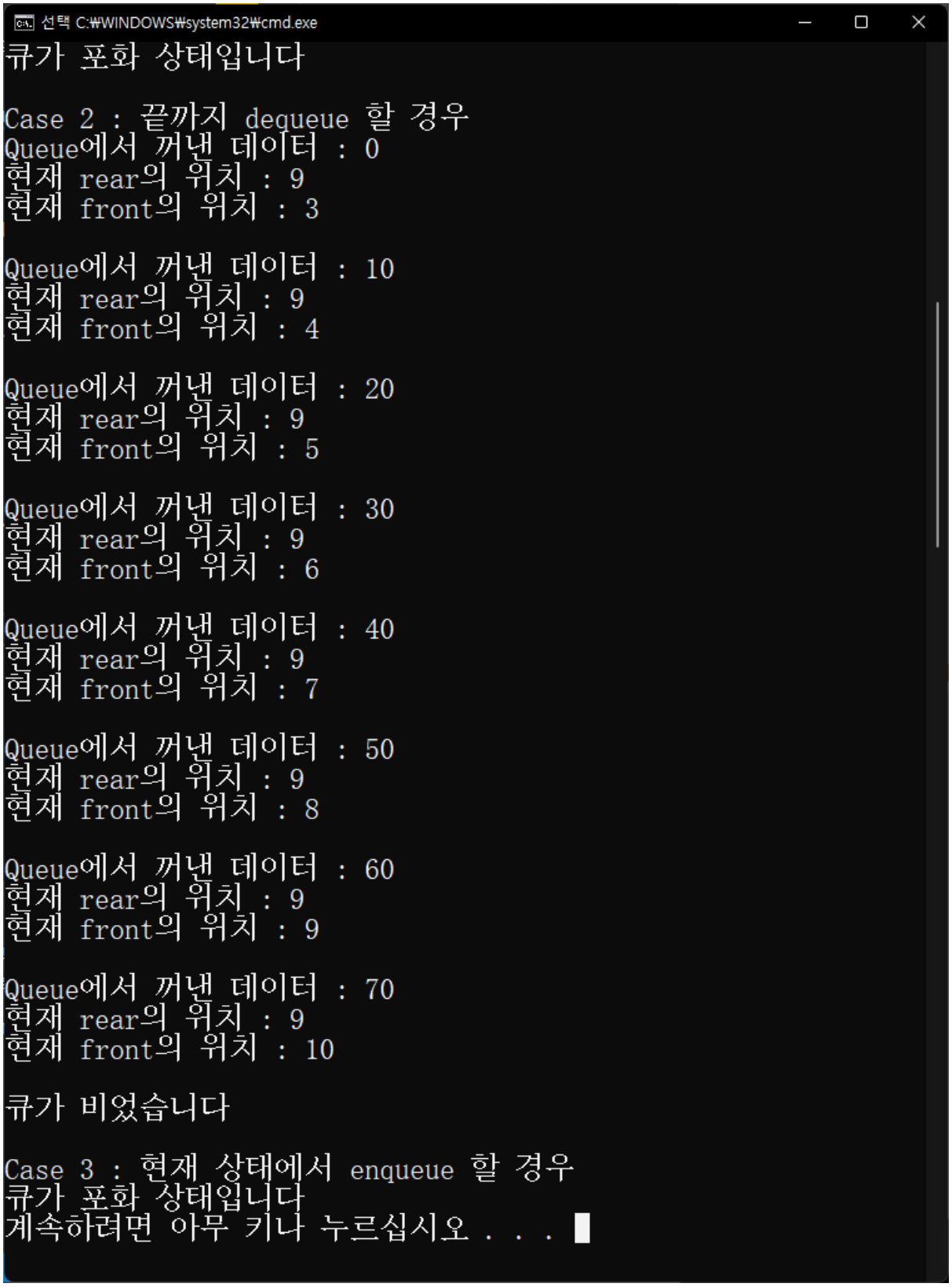
```c
#include <stdio.h>

/*
0304 Data Structures 과제 LAB 1-2
202033762 장민호

조건
1. Queue 최대크기 10
2. 구현할 function: enqueue(int), dequeue(), queue_full(), queue_empty()
3. 함수 호출 시 포인터 사용하지 않기
4. scanf, printf 는 main 에서만 사용
5. enqueue 하기 전 queue_full 검사
6. dequeue 하기 전 queue_empty 검사
7. non-Circular queue
*/

#define TRUE 1
#define FALSE 0
#define MAXSIZE 10

int front = -1;
int rear = -1;
int queue[MAXSIZE];

// Queue 의 첫 input 시 rear 과 front 를 모두 0 으로 만들어주는 과정 필요함
// 만약 front 가 rear 보다 클 경우 Queue 에는 데이터가 없음
// 만약 front 와 rear 가 모두 -1 일 경우 Queue 에는 데이터가 없음
// rear 가 MAXSIZE-1 이 되었을 때 queue_full() == YES 상태임.

void enqueue(int num) {

        if (queue_full() == FALSE) {
                if (front == -1 && rear == -1) {
                        front=0;
                }
                rear++;
                queue[rear] = num;
                printf("Queue 에 들어간 데이터 : %d\n", queue[rear]);
        }
        else {
                printf("큐가 포화 상태입니다\n");
        }
}

int dequeue() {

        if (queue_empty() == FALSE) {
                int frontData = queue[front];
```

```
                front++;
                printf("Queue 에서 꺼낸 데이터 : %d\n", frontData);
                return frontData;
        }
        else {
                printf("큐가 비었습니다\n");
        }
}

int queue_full(){
        if (rear == MAXSIZE - 1) {
                return TRUE;
        }
        return FALSE;
}

int queue_empty() {
        if (front == -1 && rear == -1) {
                return TRUE;
        }
        else if (front > rear) {
                return TRUE;
        }
        return FALSE;
}

int main() {

        enqueue(10);
        enqueue(20);
        dequeue();
        dequeue();
        dequeue();
        dequeue();

        printf("\nCase 1 : 끝까지 enqueue 할 경우\n");

        for (int num = 0; ; num += 10) {
                if (queue_full() == FALSE) {
                        enqueue(num);
                        printf("현재 rear의 위치 : %d\n", rear);
                        printf("현재 front의 위치 : %d\n\n", front);
                }
                else {
                        printf("큐가 포화 상태입니다\n");
                        break;
                }
        }

        printf("\nCase 2 : 끝까지 dequeue 할 경우\n");

        while (1) {
                if (queue_empty() == FALSE) {
                        dequeue();
                        printf("현재 rear의 위치 : %d\n", rear);
```

```
                        printf("현재 front의 위치 : %d\n\n", front);
                }
                else {
                        printf("큐가 비었습니다\n");
                        break;
                }
        }

        printf("\nCase 3 : 현재 상태에서 enqueue 할 경우\n");

        enqueue(100);

        return 0;
}
```

Screenshot


C:₩WINDOWS₩system32₩cmd.exe
Queue에 들어간 데이터 : 10
Queue에 들어간 데이터 : 20
Queue에서 꺼낸 데이터 : 10
Queue에서 꺼낸 데이터 : 20
큐가 비었습니다
큐가 비었습니다
Case 1 : 끝까지 enqueue 할 경우
Queue에 들어간 데이터 : 0
현재 rear의 위치 : 2
현재 front의 위치 : 2
Queue에 들어간 데이터 : 10
현재 rear의 위치 : 3
현재 front의 위치 : 2
Queue에 들어간 데이터 : 20
현재 rear의 위치 : 4
현재 front의 위치 : 2
Queue에 들어간 데이터 : 30
현재 rear의 위치 : 5
현재 front의 위치 : 2
Queue에 들어간 데이터 : 40
현재 rear의 위치 : 6
현재 front의 위치 : 2
Queue에 들어간 데이터 : 50
현재 rear의 위치 : 7
현재 front의 위치 : 2
Queue에 들어간 데이터 : 60
현재 rear의 위치 : 8
현재 front의 위치 : 2
Queue에 들어간 데이터 : 70
현재 rear의 위치 : 9
현재 front의 위치 : 2
큐가 포화 상태입니다


선택 C:₩WINDOWS₩system32₩cmd.exe
큐가 포화 상태입니다
Case 2 : 끝까지 dequeue 할 경우
Queue에서 꺼낸 데이터 : 0
현재 rear의 위치 : 9
현재 front의 위치 : 3
Queue에서 꺼낸 데이터 : 10
현재 rear의 위치 : 9
현재 front의 위치 : 4
Queue에서 꺼낸 데이터 : 20
현재 rear의 위치 : 9
현재 front의 위치 : 5
Queue에서 꺼낸 데이터 : 30
현재 rear의 위치 : 9
현재 front의 위치 : 6
Queue에서 꺼낸 데이터 : 40
현재 rear의 위치 : 9
현재 front의 위치 : 7
Queue에서 꺼낸 데이터 : 50
현재 rear의 위치 : 9
현재 front의 위치 : 8
Queue에서 꺼낸 데이터 : 60
현재 rear의 위치 : 9
현재 front의 위치 : 9
Queue에서 꺼낸 데이터 : 70
현재 rear의 위치 : 9
현재 front의 위치 : 10
큐가 비었습니다
Case 3 : 현재 상태에서 enqueue 할 경우
큐가 포화 상태입니다
계속하려면 아무 키나 누르십시오 . . .