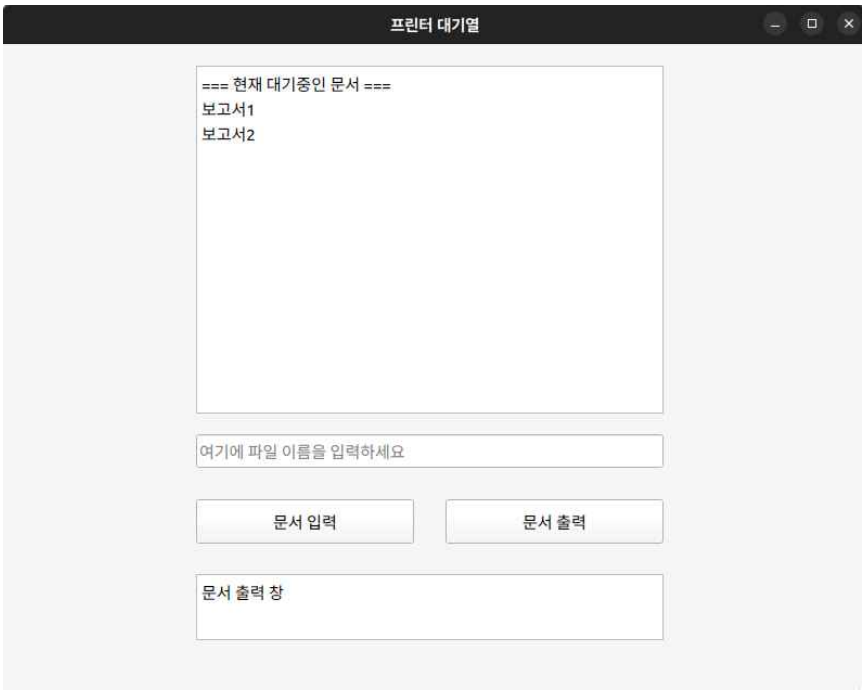
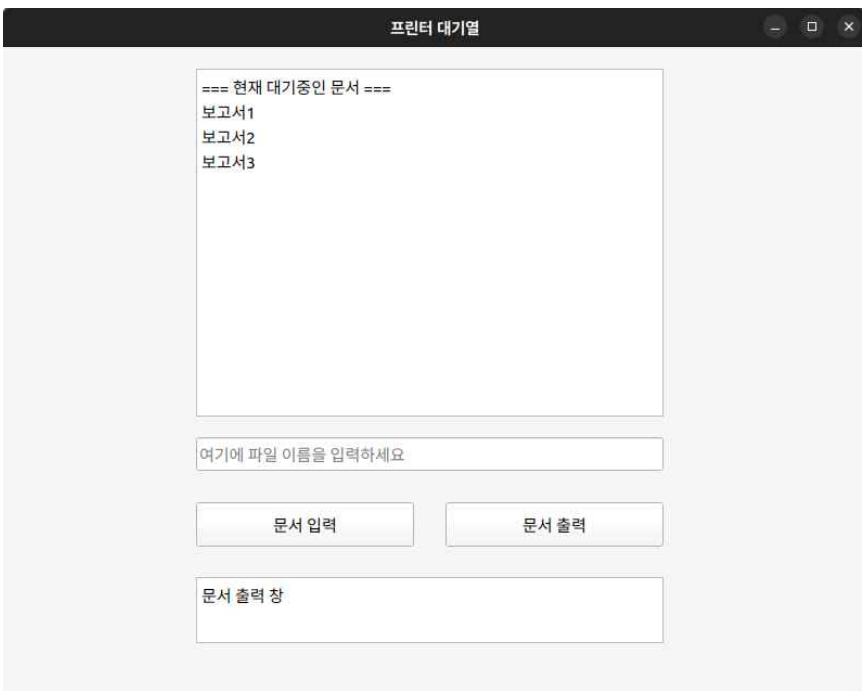
QT 보고서

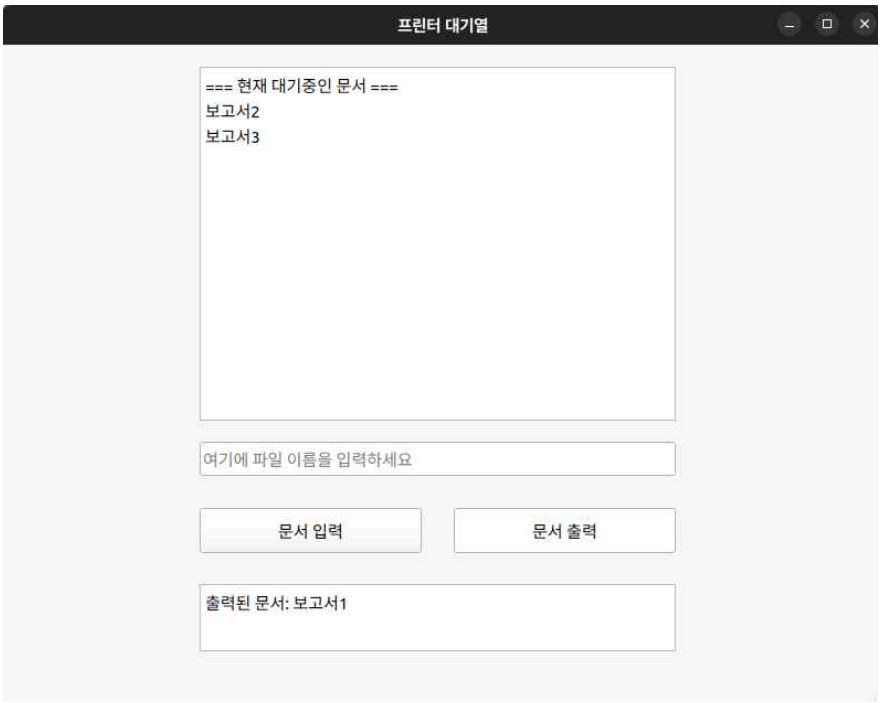
20기 인턴 송수민

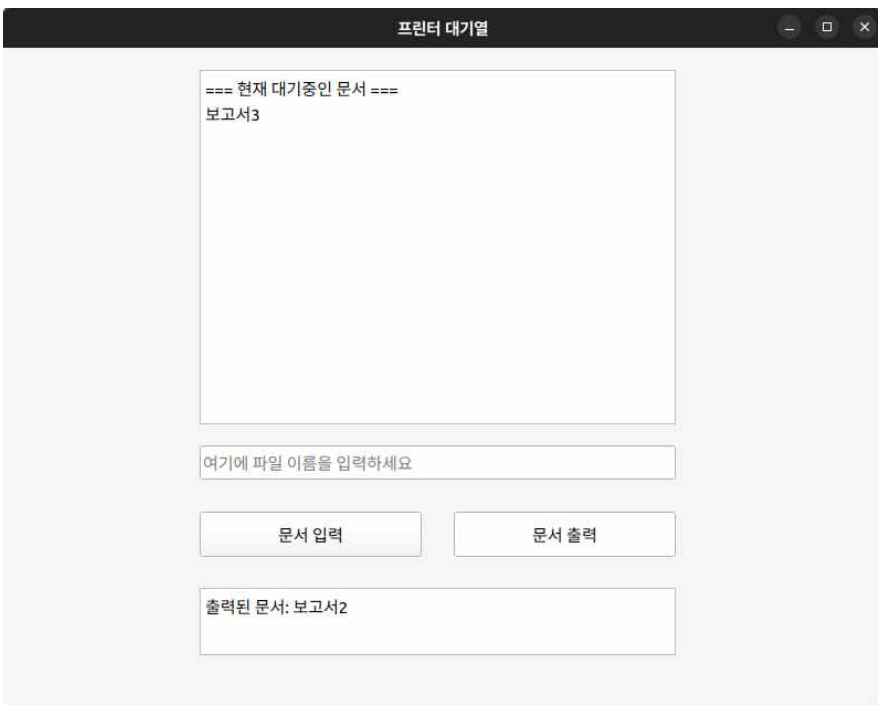
-hw2

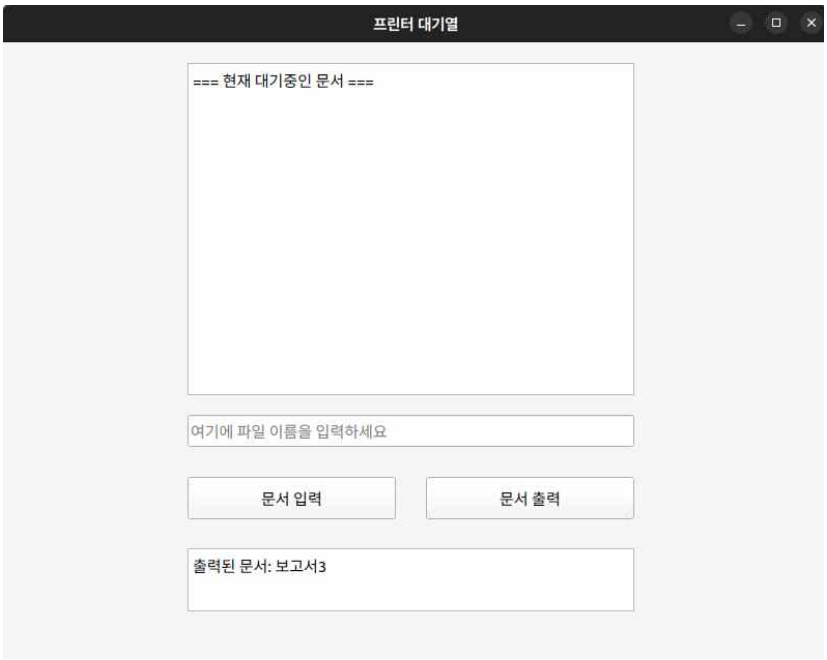
1. 실행 결과

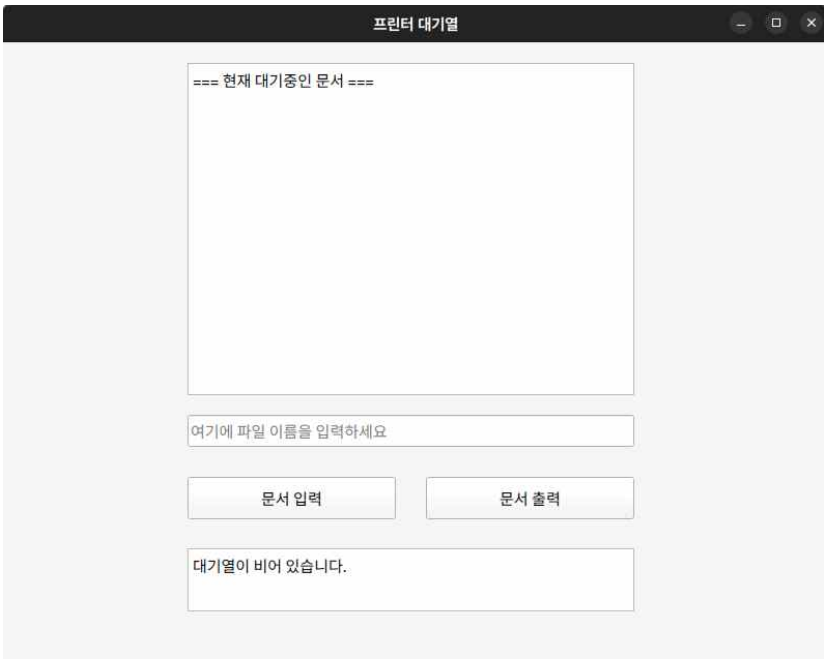





프린터 대기열
=== 현재 대기중인 문서 ===
보고서2
보고서3
여기에 파일 이름을 입력하세요
문서 입력
문서 출력
출력된 문서: 보고서1


프린터 대기열
=== 현재 대기중인 문서 ===
보고서3
여기에 파일 이름을 입력하세요
문서 입력
문서 출력
출력된 문서: 보고서2


프린터 대기열
=== 현재 대기중인 문서 ===
여기에 파일 이름을 입력하세요
문서 입력
문서 출력
출력된 문서: 보고서3


프린터 대기열
=== 현재 대기중인 문서 ===
여기에 파일 이름을 입력하세요
문서 입력
문서 출력
대기열이 비어 있습니다.

2. 코드 분석

1) 헤더 파일

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <vector>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void enqueue();       // 문서 입력
    void dequeue();       // 문서 출력
    void updateDisplay(); // 대기열 표시 갱신

private:
    Ui::MainWindow *ui;
    std::vector<QString> queue; // Queue 자료구조
};
#endif // MAINWINDOW_H
```

파일을 입출력할 클래스를 만들고 각각의 함수와 벡터로 Queue 자료형으로 먼저 입력한 문서가 먼저 출력되도록 프린터기 대기 프로그램을 만들기 위한 선언을 함

2) mainwindow.cpp 파일

```cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->input_file_name->setPlaceholderText("여기에 파일 이름을 입력하세요");


    connect(ui->input_button, &QPushButton::clicked, this, &MainWindow::enqueue);
    connect(ui->out_button, &QPushButton::clicked, this, &MainWindow::dequeue);

    ui->name_list->setReadOnly(true);
    ui->output_name->setReadOnly(true);
    ui->name_list->setText("=== 현재 대기중인 문서 ===");
    ui->output_name->append("문서 출력 창");
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::enqueue()
{
    QString doc = ui->input_file_name->text();
    if (!doc.isEmpty()) {
        queue.push_back(doc); // 큐 끝에 추가
        ui->input_file_name->clear();
        updateDisplay();
    }
}
```

```cpp
void MainWindow::dequeue()
{
    if (!queue.empty()) {
        // 맨 앞 문서 가져오기
        QString printedDoc = queue.front();
        queue.erase(queue.begin()); // FIFO 제거

        // 출력창에 표시 (이전 출력 내용은 덮어씀)
        ui->output_name->clear();
        ui->output_name->append("출력된 문서: " + printedDoc);

        updateDisplay();
    } else {
        ui->output_name->setText("대기열이 비어 있습니다.");
    }
}

void MainWindow::updateDisplay()
{
    ui->name_list->clear();
    ui->name_list->append("=== 현재 대기중인 문서 ===");

    for (const auto &doc : queue) {
        ui->name_list->append(doc);
    }
}
```

3 - 13줄 : MainWindow 생성자, 부모 클래스(QMainWindow)를 초기화하면서 UI 객체(ui)를 세팅. 입력창에 placeholder 텍스트를 지정하고, 버튼 클릭 시 enqueue·dequeue 슬롯 함수가 호출되도록 connect로 시그널-슬롯을 연결함. 출력용 QTextEdit은 읽기 전용으로 설정하고 초기 안내 문구를 출력함.

15 - 18줄 : 소멸자(MainWindow::~MainWindow), new로 할당된 ui 객체를 해제하여 메모리 누수 방지.

20 - 29줄 : enqueue 함수. 입력창에서 텍스트를 가져와 비어 있지 않으면 queue의 끝에 추가(push_back). 이후 입력창을 비우고, 대기열 목록을 새로 그리기 위해 updateDisplay() 호출.

31 - 46줄 : dequeue 함수. 큐가 비어 있지 않으면 맨 앞 문서를 꺼내어 출력창에 표시 후 삭제(FIFO 구조). 큐가 비어 있으면 “대기열이 비어 있습니다.”라는 메시지를 출력. 마지막에 updateDisplay() 호출로 목록 갱신.

48 - 56줄 : updateDisplay 함수. 출력창(name_list)을 지우고 헤더 문구를 다시 출력한 뒤, 현재 queue에 들어 있는 문서들을 순서대로 append하여 화면에 표시.

3) main.cpp 파일

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.setWindowTitle("프린터 대기열");
    w.show();
    return a.exec();
}
```

창 이름 설정과 각각의 함수들로 프로그램을 작동시켜준다.