COMP9032 Project Report 2024 T3

z5325184 Changyue Tan
z5489849 Ating Chen
z5547906 Yinan Cai
z5509512 Jingyuan Xue
z5548312 Jinchuan He

Product Manual

Introduction

The development board and assembly software prototype a self-service kiosk for patient management in a small medical clinic. It streamlines patient registration, queuing, and doctor consultations by enabling self-check-in, appointment management, and wait time reminders on LED bar and audible alerts from a bell. Using LCD displays, LED panels, and buttons, the system automates patient queuing and appointment scheduling.

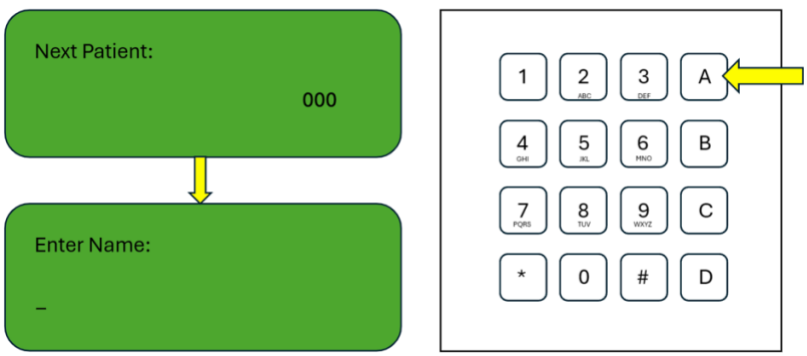
Hardware Components and Operation Instructions

Main LCD: Displays interactive prompts, patient input, and queue numbers. Dynamically updates in "Entry Mode" to show patient name as it is being entered.

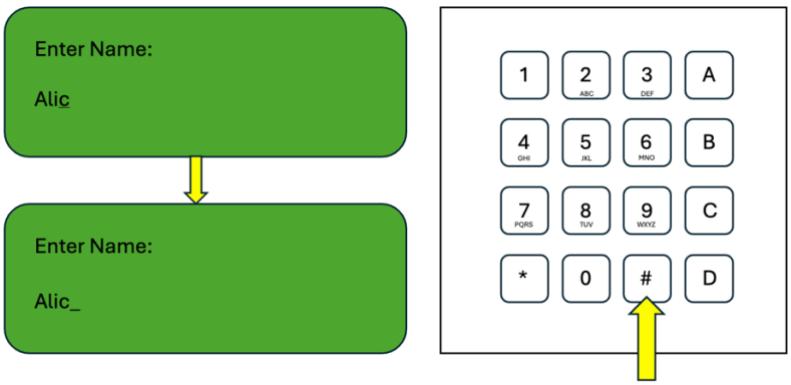
Secondary LCD: Display the main LCD content in “Display Mode”. Maintains previous content during “Entry Mode” to provide next patient information to the waiting patients.

4x4 Matrix Keypad: Used for patient name input, backspace, delete, and confirmation. Supports switching between “Display” and “Entry” modes. When switching to "Display Mode", all inputs should be ignored.

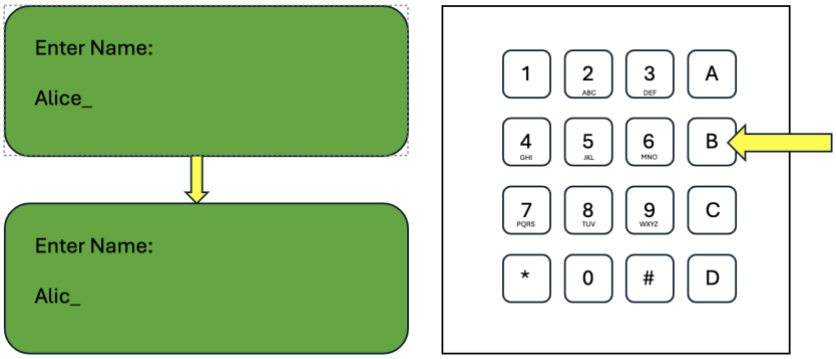
- Numeric Keys (2-9): Simulate traditional mobile phone input, where each numeric key corresponds to multiple letters (e.g., key 2 for A, B, C).
- ‘A’ Key: Start Entry Mode



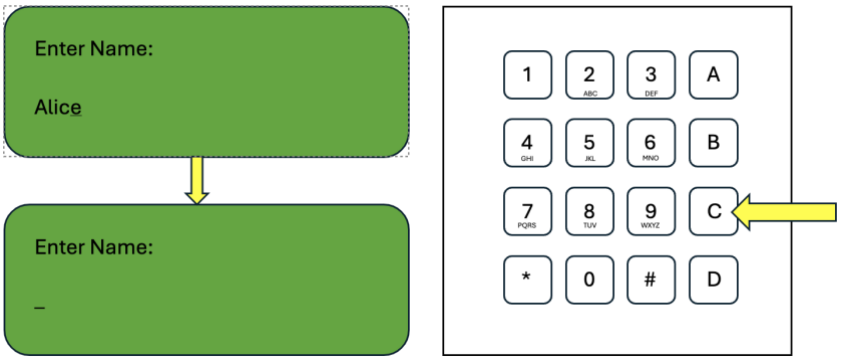
- ‘#’ Key: Temporarily save the character entered by the current patient.



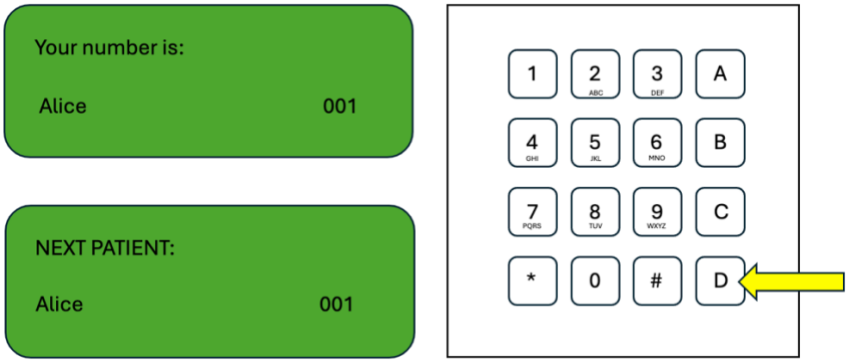
- ‘B’ Key (Backspace): Deletes the most recently entered (temporarily saved) character. Note: Backspace can only be pressed after current letter is confirmed (‘#’ key pressed), else the previous letter will be removed instead of the current one.



- ‘C’ Key (Clear): Clears all input.



- ‘D’ Key (Done): Confirms the input name and assigns a queue number to the patient.



LED Panel: Provides visual alerts with the following features:

- Estimated remaining time: The number of LEDs to be turn on correspond to the proportion of time remaining out of the estimated total consultation time. If the consultation is halfway through, 5 out of the 10 LED bars will be turn on.
    - 70% time remaining

    - 40% time remaining

- Patient Call Alerts: All LED bars flash in sync with the tone of the motor (bell) to draw attention from of patient being called. The flashes and tone will become more rapidly if the patient does not enter the doctor's room within 10 seconds.

Motor: Generates ring tones of a bell in sync with flashing LED bars

- Beep-beep: A 1-second ON and 1-second OFF pattern to call the next patient
- Bip-bip: A 0.5-second ON and 0.5-second OFF pattern 10 seconds after first pattern
- Bee...eep: A 3-second continuous tone to signal appointment is cancelled

Push Buttons:

- PB0 Button (Next Patient): Once pressed, calls the next patient. If the patient arrives to the doctor's room. The doctor can press and hold the button to silence the bell. And the next patient will be displayed on to both LCDs. The LCD of the kiosk will return to its original state after 5 seconds, if someone was using the kiosk before the doctor pressed the next patient button.
  Note: If no patients are in the queue, any press on next patient button is ignored.
- PB1 Button (Cancel Appointment): If no one comes into the doctor's room, the doctor can choose to cancel the appointment for that patient, by press and hold this button for 1 seconds. The bell and LED will ring and lid up for 3 seconds, and the next patient will be called, while their name and number being displayed on the LCDs.
  Note: If there is no more people waiting in the queue, the patient can still cancel the appointment of the current patient can starts calling the next patient. The LCD will essentially display and empty name with patient number. The issue was not properly addressed due to time constraints. If more time was allowed for the development of the prototype, it is easily solved by adding a checker after the cancellation of the current patient and before calling the next patient. If there is no more people in the queue, then just do the cancellation, do not call the next patient. Otherwise, do the cancellation, and then call the next patient.

Appendix

Project timeline

The project took approximately 3 weeks from start to finish, with 1 week of high-level design, 1 week of low-level implementation, and 1 week of on-board testing. Detailed timeline and relevant workload distribution among team members are as follows:

- **10/10/2024**: Project Released

- **26/10/2024**: Face to Face meeting on Campus (3 hours):
  - First meeting
  - Discussed Project specification and requirement
  - Post questions about possible typos in assignment specification
  - Clarified doubts in project spec with course staff
  - Get to familiar with teammates and everyone's capabilities

- **2/11/2024**: Online meeting on MS Teams (1 hour):
  - Determined how to split Project's workload
  - Ating Chen:
    - Implement Keypad Functions
    - Implement Queue data structure in memory
    - Record input of patient names
    - Assign each patient a patient number
  - Jingyuan Xue:
    - LCD Display mode
    - LCD Entry mode
  - Yinan Cai:
    - LED flash patterns
  - Changyue Tan:
    - Team collaboration support, code merge and documentation
    - Project backbone, main program development
    - High level execution flow design
    - Low level subroutine and macro implementation and testing

- **6/11/2024**: Face to Face meeting during weekly lab session (2 hours):
  - New team member joined: Jinchuan He
  - Shift workload around teammates based on different capabilities dedicated time commitments for the project
  - Ating Chen:
    - Implement Keypad functions (in progress)
    - Implement Queue data structure in memory (moved to Changyue Tan)

        - Record input of patient names (to Changyue Tan)
        - Assign each patient a patient number (to Changyue Tan)
    - Jingyuan Xue:
        - LCD Display mode (in progress)
        - LCD Entry mode (in progress)
    - Yinan Cai:
        - LED flash patterns (completed, based on static delays)
        - Bell/Motor ringing patterns (new task added during meeting)
    - Changyue Tan:
        - Project backbone, main program development (in progress)
        - High level execution flow design (completed)
        - Low level subroutine and macro implementation and testing (completed, provides interface to hardware for other parts of the program)
        - Implement Queue data structure in memory (moved from Ating Chen)
        - Record input of patient names (moved from Ating Chen)
        - Assign each patient a patient number (moved from Ating Chen)
        - Timer Interrupts (new task added during meeting)
    - Jinchuan He (new team member):
        - LED Display of remaining time till next patient

- **9/11/2024**: Face to Face meeting on Campus (4 hours):
    - Final meeting. Update on task distributions.
    - Ating Chen:
        - Implement Keypad functions (completed, successfully compiles, but does not work on lad board)
    - Jingyuan Xue:
        - LCD Display mode (Completed)
        - LCD Entry mode (Moved to Changyue Tan for interaction with keypad functions)
        - Push button interrupts (New task added during meeting)
    - Yinan Cai:
        - LED flash and Bell ring tone patterns (completed, based on static delays)
    - Changyue Tan:
        - Project backbone, main program development (Completed)
        - Low level subroutine and macro implementation and testing (completed)
        - Implement Queue data structure in memory (Completed)
        - Record input of patient names (In progress)
        - Assign each patient a patient number (In progress)
        - Timer Interrupts (Completed)
    - Jinchuan He (new team member):
        - LED Display of remaining time till next patient (In progress)

- **11/11/2024**: No meeting, group chat communication
    - Changyue Tan

        - Keypad functions (completed)
        - LCD Entry mode (completed, linked with Keypad functions)
        - LED flash and Bell ring tone linked with timer interrupt
    - Jingyuan Xue
        - Push button interrupts (Completed, as a standalone falling edge trigger interrupt)

- **12/11/2024:** No meeting, group chat communication
    - Jinchuan He
        - LED bar remaining time display (incomplete, not linked to remaining time variable in data memory space)
    - Changyue Tan
        - Push buttons linked with dequeue and enqueue
        - Entry Mode and Display Mode linked with push buttons
        - LED bar remaining time display
        - Nest Interrupt logic
        - Detect push button 1 second hold during interrupt
        - Graceful return from Display Mode to Entry Mode
        - Strobe LED to distinguish LCD communication.
        - Final testing on board

- **13/11/2024**: Project Demo Day

Design Choice Justifications

- **Patients queue**:
    - A custom defined Queue Data Structure and its relevant interfaces are designed for easier control over enqueues and dequeues. All interactions with memory and variables related to the queue are maintained and encapsulated by this Abstract Data Type. Other functions can access the queue using only a number of functions or variables. This lowers the burden of directly manipulating the queue from other parts of the program.

- **Keypad**:
    - Originally, letters are retried by implemented ‘if’ statements, which turns to be overly complex. Hence, we switched to the 2D array approach.
    - Alphabetical letter literals are stored in program memory together with the total number of letters can be referred from a key. This provides an easy way to know when to wrap around the index to 0.
    - ‘#’ acts like an “enter” on a normal keyboard. This gives a more precise way of confirmation of the entry of a letter to eliminate ambiguity on how much time should the system wait until the letter is considered entered, while also gives patient more time to respond or think.

- **Calling for next patient:**
  - When next patient button is pushed, an external interrupt is triggered. All functionalities while the program is in this “calling next patient” state is executed from within this interrupt. By doing so we can ensure that when the program returns from “calling next patient” state, it will be returned to whatever mode it was in before this “calling next patient” external interrupt is triggered. In this way, the control flow can be gracefully handed over to Entry Mode or Display Mode without disrupting the original execution flow.

- **Time counter and timer interrupts**:
  - Originally all functions related to the passage of time is using nop delay, which prevented other parts of the program, or interrupts from executing.
  - Several time counter are implemented in data memory space, which are updated during timer interrupts at certain time intervals. This allows different parts of the program to be aware of the passage of time, and implement time related functionalities without using nop delay.

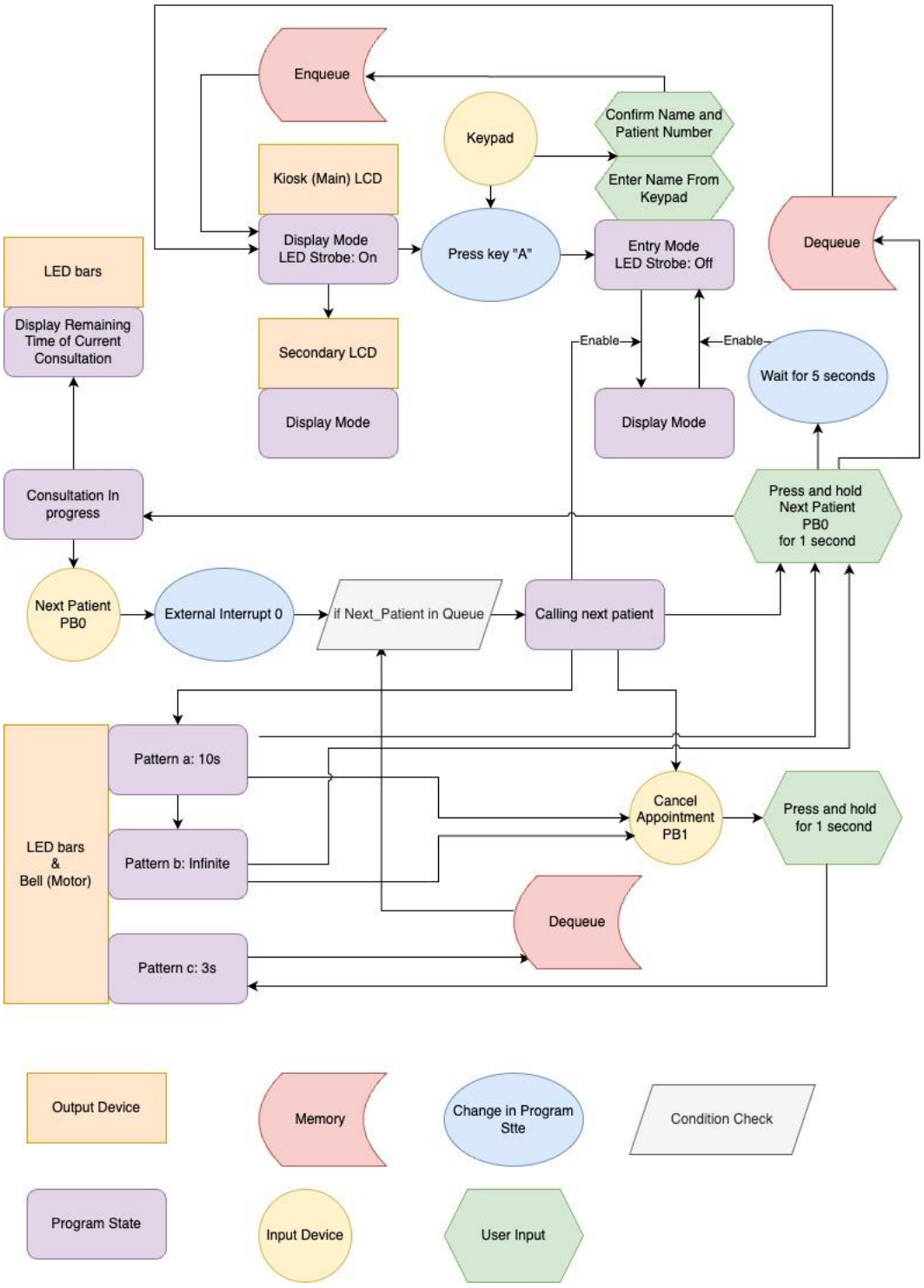
- **Exit from LED flash pattern and bell ring tone**:
  - Originally a function, or subroutine, was used to check if either push buttons are pushed. However, if a button is pushed, it jumps to other parts of the code and does not return. Hence, several rjmps were used instead of rcall.

Pseudo code of Overall Design

```
char temp_name[10];
char Patient_Name[10];
char Patients_Queue[2560];
char *Next_Patient = Patients_Queue;
char *Last_Patient = Patients_Queue + 10;
char *Space_For_New_Patient = Patients_Queue + 10;
int Next_Patient_Number = 0;
int Last_Patient_Number = 0;
long Temp_Counter = 0;
long Seconds_Counter = 0;
int Blink_Timer = 0;
int Entry_Mode_Flag = 0;
int Entry_Confirm_Flag = 0;
char *Entry_Mode_Prompt = "Enter Name";
char *Entry_Mode_Complete_Message = "Your Number Is";
char *Display_Mode_Message = "Next Patient";
long key_offsets[8] = { &key2_letters, &key3_letters,
                        &key4_letters, &key5_letters,
                        &key6_letters, key7_letters,
                        &key8_letters, &key9_letters,
                        }
struct keyN_letters {
    int number_of_letter_can_be_index_by_this_key = 3;
    char *letter_literals_of_this_key = {'A', 'B', 'C'};
}
...
Timer_interrupt_0() {
    Temp_Counter ++;
    if (Temp_Counter == 500) {
        Blink_Timer++;
    } else if (Temp_Counter == 1000) {
        Blink_Timer++;
        Seconds_Counter++;
        Temp_Counter = 0
    }
    display remain time to LED bar
}
```

```
Interrupt_0():
    Start_of_interrupt:
    if (no next patient) {return from interrupt;}
    display_next_patient();
    while (1) {
        if (time elapsed < 10) {do pattern_a;}
        else {do pattern_b;}
        if (cancelation or patient arrives) {break;}
    }
    if (patient arrives) {goto end_of_interrupt;}
    else if (cancelation) {
        do pattern C
        goto start_of_interupt
    }
    end of interrupt:
    if (Entry_Mode_Flag) {
        print(entry mode msg to LCD);
    } else if (Entry_Confirm_Flag) {
        Print(entry confirm msg to LCD);
    }
Main(){
    while (1) {
        display_next_patient();
        take_input_from_keypad();
        if (keypad input == 'A'){
            entry_mode(); continue;
        }
    }
}
entry_mode() {
    Entry_Mode_Flag = 1;
    while not ('D' is pressed) {
        take_input_from_keypad();
        print(whatever is entering by patient)
        if ('D' is pressed) {break;}
    }
    Entry_Mode_Flag = 0;
    Entry_Confirm_Flag = 1;
    print(Entry_Mode_Complete_Message);
    Entry_Confirm_Flag = 0;
    Return;
}
```

Execution Flow Block Diagram



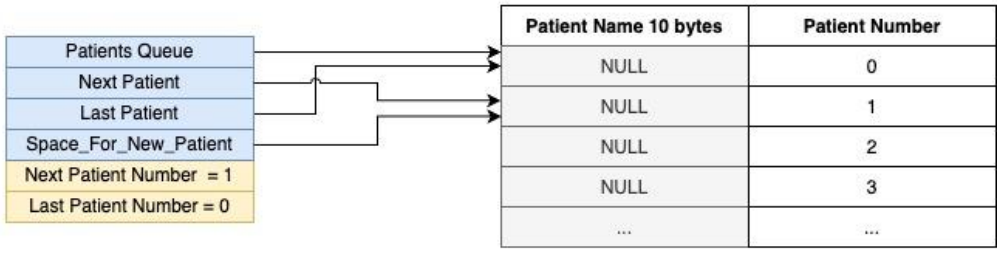
Queue Data Structure & Interface

Initial State:
Next_Patient = Patients_Queue + 10
Last_Patient = Patients_Queue
Space_For_New_Patient = Patients_Queue + 10

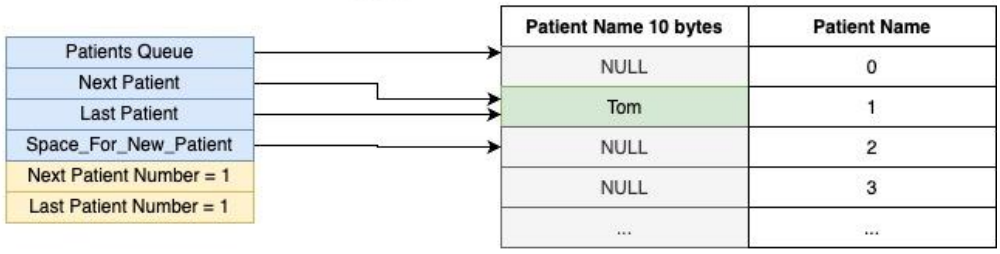
Enqueue:
Last_Patient += 10
Last_Patient_Number += 1
Space_For_New_Patient += 10

Dequeue:
Next_Patient += 10
Next_Patient_Number += 1

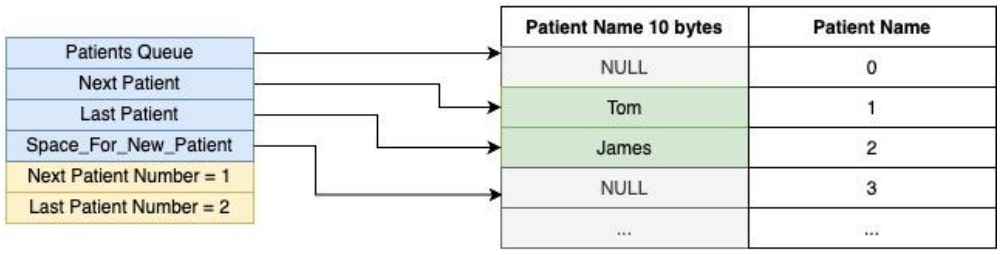
Initialise



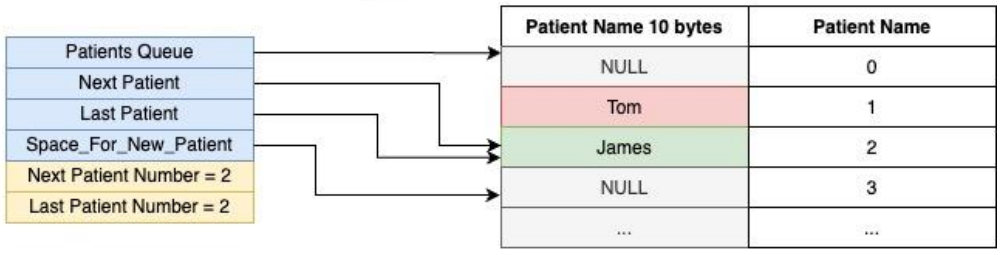
Enqueue



Enqueue



Dequeue



Dequeue