Microprocessor hospital prject

Sarah alrefaey abdullah 2020/02308

Maria ayman Ibrahim 2020/09891

➢ **Project over view:**

The provided code implements a simple hospital management system in C. It allows users to log in with different privileges such as admin, regular user, and patient. Depending on the privilege level, users can perform various operations including updating patient data, removing reservations, checking empty rooms, and logging out.

➢ **Purpose and scope:**

The purpose of this project is to develop a basic hospital management system to facilitate tasks such as managing patient data, room reservations, and user authentication. The system allows users to perform different operations based on their privileges. The scope of the project includes user authentication, patient management, room management, and basic user interactions.

➢ **Features:**

1.User authentication:

The system has a user authentication module where users can log in using their passwords. There are three types of users in the system: admin, regular user, and patient

2.Room management:

The system can manage hospital rooms. It can check for empty rooms and display them to the user. It also keeps track of reserved rooms.

3.**patient management**:

The system can manage patient data. It can add new patients, update patient data, and remove patient reservations. The system stores the patient's ID, name, and status (good or bad).

➢ **Functions:**

The code is structured into several functions and utilizes structures to store user, room, and patient data

1. **User/room/patient** they are structures store user, room and patient data.
2. **Login** handles user authentication and redirects users to appropriate functions based on their privileges
3. **Update_patient_data** allows user to update patient data such as named and statues ‘0 for bad status and 1 for good status’
4. **Remove_current_reservation** Allows users to remove reservations for currently reserved rooms.
5. **Create new patient**: Enables users to create new patient records.
6. **Check_empty_rooms:** Displays the list of empty rooms.
7. **Check_current_reservation**: Displays the list of currently reserved rooms.

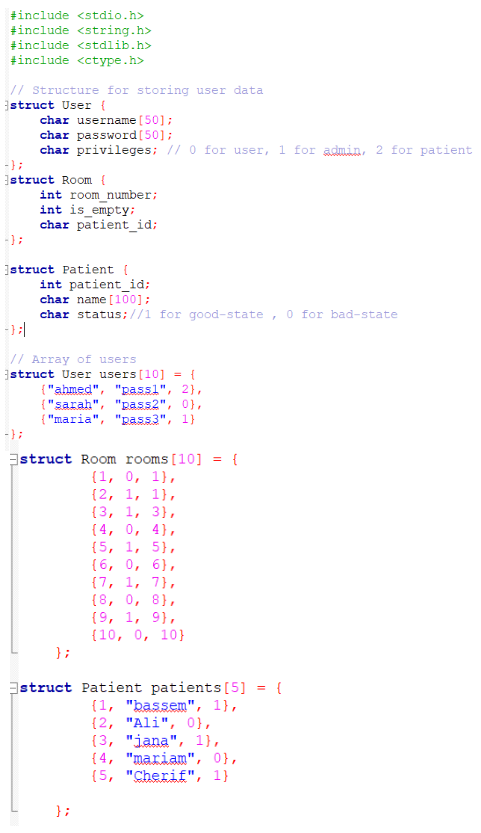
➢ 1st part the libraries and structures:

- **Header file and library:**

1.#include<stdio.h>: this library is for input and output operation.
2.#include<stdlib.h>: this library is for memory allocations.
3.#include<string.h>: this library is for manipulating strings.
4. #include<ctype.h>: this library is for providing functions and transforming characters.

- **Structures:**
  *This code contains 3 structures to define user, room and patient that saves data needed in the code

structure	Field	description
user	1)User name: 2)Password: 3)Privilege:	1)A string of up to 50 characters representing the user's username 2) A string of up to 50 characters representing the user's password 3) A character representing the user's privileges: 0 for regular user, 1 for admin, 2 for patient
room	1)room_number 2)is_empty 3)patient_id	1) An integer representing the room number 2) A boolean indicating whether the room is empty or not (1 for empty, 0 for occupied) 3) A character representing the ID of the patient currently occupying the room (if any)
patient	1)name 2)status	1) A string of up to 100 characters representing the patient's name 2) A character representing the patient's status: 1 for good-state, 0 for bad-state



```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

// Structure for storing user data
struct User {
    char username[50];
    char password[50];
    char privileges; // 0 for user, 1 for admin, 2 for patient
};
struct Room {
    int room_number;
    int is_empty;
    char patient_id;
};

struct Patient {
    int patient_id;
    char name[100];
    char status;//1 for good-state , 0 for bad-state
};

// Array of users
struct User users[10] = {
    {"ahmed", "pass1", 2},
    {"sarah", "pass2", 0},
    {"maria", "pass3", 1}
};
struct Room rooms[10] = {
        {1, 0, 1},
        {2, 1, 1},
        {3, 1, 3},
        {4, 0, 4},
        {5, 1, 5},
        {6, 0, 6},
        {7, 1, 7},
        {8, 0, 8},
        {9, 1, 9},
        {10, 0, 10}
    };

struct Patient patients[5] = {
        {1, "bassem", 1},
        {2, "Ali", 0},
        {3, "jana", 1},
        {4, "mariam", 0},
        {5, "Cherif", 1}

    };
```

part-1

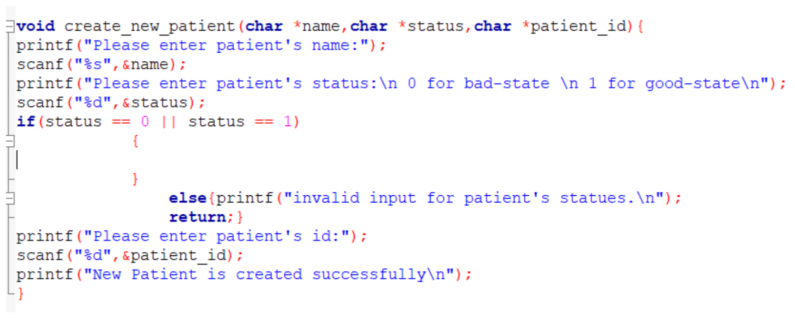
*Each structure defined by array of maximum number:

-Struct User users include [10]

-struct Room rooms include [10]

-struct Patient patients include [5]

➢ 2nd create_new_patient function:



```c
void create_new_patient(char *name,char *status,char *patient_id){
printf("Please enter patient's name:");
scanf("%s",&name);
printf("Please enter patient's status:\n 0 for bad-state \n 1 for good-state\n");
scanf("%d",&status);
if(status == 0 || status == 1)
              {

              }
                   else{printf("invalid input for patient's statues.\n");
                   return;}
printf("Please enter patient's id:");
scanf("%d",&patient_id);
printf("New Patient is created successfully\n");
}
```

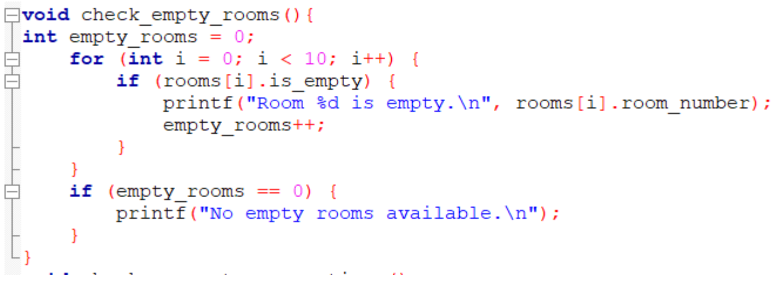
part-2

-The function prompts the user to enter a patient's name, status and ID and they are identified by character

- the user entre the patient name and prints it then followed same by ID

-then for status it has choices' 0' for good status and '1' for bad status otherwise prints invalid input for patient's status .

➢ 3rd check_empty _rooms function:



```c
void check_empty_rooms(){
int empty_rooms = 0;
    for (int i = 0; i < 10; i++) {
        if (rooms[i].is_empty) {
            printf("Room %d is empty.\n", rooms[i].room_number);
            empty_rooms++;
        }
    }
    if (empty_rooms == 0) {
        printf("No empty rooms available.\n");
    }
}
```

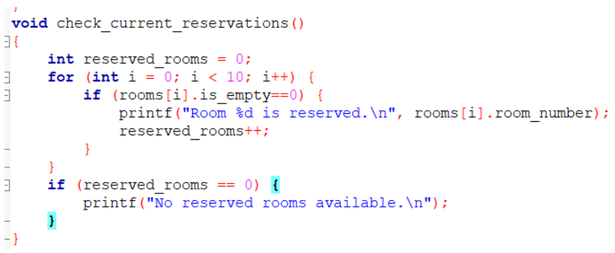
part-3

- the function checks for empty rooms in the rooms array and prints out the room numbers that are empty.

- It initializes a counter variable **empty_room** to 0 and then iterates through the rooms array.

-if the condition is true it prints the room number and increment **empty_room** counter and if it's not it prints no empty rooms available.

4th check_current_reservation function:



```
void check_current_reservations()
{
    int reserved_rooms = 0;
    for (int i = 0; i < 10; i++) {
        if (rooms[i].is_empty==0) {
            printf("Room %d is reserved.\n", rooms[i].room_number);
            reserved_rooms++;
        }
    }
    if (reserved_rooms == 0) {
        printf("No reserved rooms available.\n");
    }
}
```

part4

-the function checks for reserved rooms in rooms array and prints out their room number

-we initialize variable **reserved_rooms** to 0 and iterates through rooms array if a room is not empty it prints the rooms number and increment the **reserved_rooms** counter

-is still **reserved_room** is still 0 it prints message indicating to reserved rooms available.

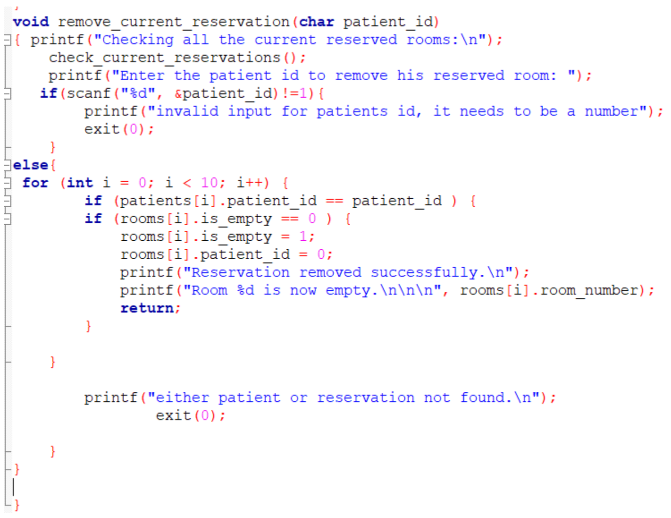
5th remove_current_reservation function:

-the function is used to remove the reservation of a patient from a room in the room array

-it first points out all the currently reserved rooms using **check_current_reservation** function then it prompts the user to enter the patient ID that will be removed

-if the input is not integer it prints and error message and exists, If the patient ID is found in the patient's array, the function checks if the corresponding room in the rooms array is not empty. If it is not empty, the **is_empty** field is set to 1 and the **patient_id** field is set to 0, indicating that the room is now empty and prints a success message

-either patient ID or reservation not found it prints error message.



```
void remove_current_reservation(char patient_id)
{ printf("Checking all the current reserved rooms:\n");
    check_current_reservations();
    printf("Enter the patient id to remove his reserved room: ");
   if(scanf("%d", &patient_id)!=1){
        printf("invalid input for patients id, it needs to be a number");
        exit(0);
    }
else{
 for (int i = 0; i < 10; i++) {
        if (patients[i].patient_id == patient_id ) {
        if (rooms[i].is_empty == 0 ) {
            rooms[i].is_empty = 1;
            rooms[i].patient_id = 0;
            printf("Reservation removed successfully.\n");
            printf("Room %d is now empty.\n\n\n", rooms[i].room_number);
            return;
        }

    }

        printf("either patient or reservation not found.\n");
                exit(0);

    }
}
|
}
```

part5

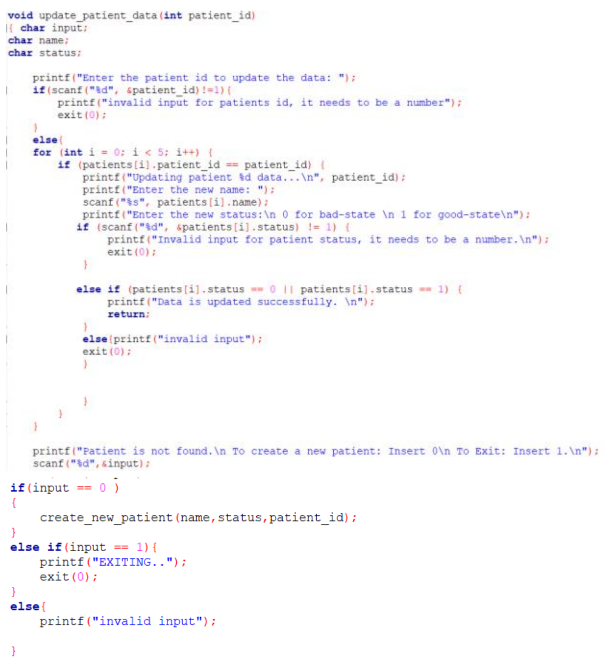
6th update patient data function:

-the function is used to update the data of patient in the data , it first prompts the user to entre patient ID that will be updated.

-if the input is not a number an error message is displayed and exists , If the patient ID is found in the patients array, the function prompts the user to enter the new name and status of the patient

-if it’s valid the function updates the corresponding name and status fields in the patients array and prints a success message

-if it’s not the function choices the user either to exist or create a new patient.

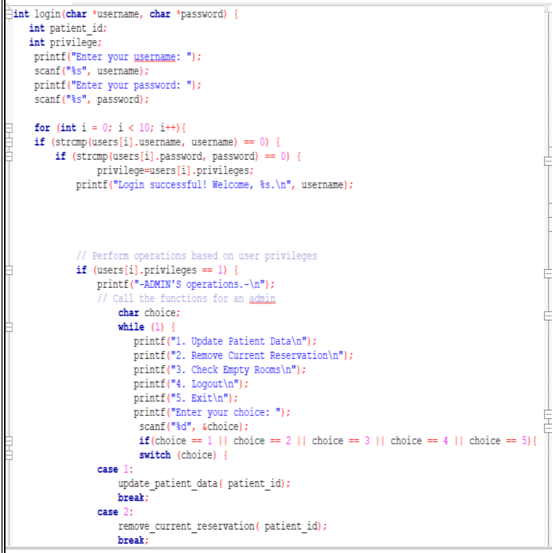


```c
void update_patient_data(int patient_id)
{ char input;
char name;
char status;

    printf("Enter the patient id to update the data: ");
    if(scanf("%d", &patient_id)!=1){
        printf("invalid input for patients id, it needs to be a number");
        exit(0);
    }
    else{
    for (int i = 0; i < 5; i++) {
        if (patients[i].patient_id == patient_id) {
            printf("Updating patient %d data...\n", patient_id);
            printf("Enter the new name: ");
            scanf("%s", patients[i].name);
            printf("Enter the new status:\n 0 for bad-state \n 1 for good-state\n");
           if (scanf("%d", &patients[i].status) != 1) {
                printf("Invalid input for patient status, it needs to be a number.\n");
                exit(0);
            }

           else if (patients[i].status == 0 || patients[i].status == 1) {
                printf("Data is updated successfully. \n");
                return;
            }
            else{printf("invalid input");
            exit(0);
            }


            }
        }
    }

    printf("Patient is not found.\n To create a new patient: Insert 0\n To Exit: Insert 1.\n");
    scanf("%d",&input);
if(input == 0 )
{
    create_new_patient(name,status,patient_id);
}
else if(input == 1){
    printf("EXITING..");
    exit(0);
}
else{
    printf("invalid input");

}
```

part6

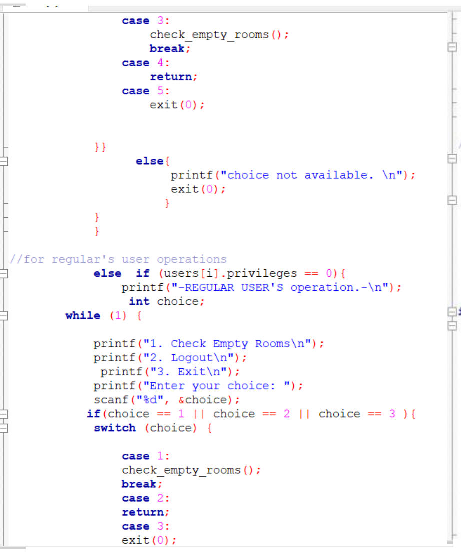
7th login function:

Login function includes most of the operations in the program and done after the rest function to be able to call them in login function



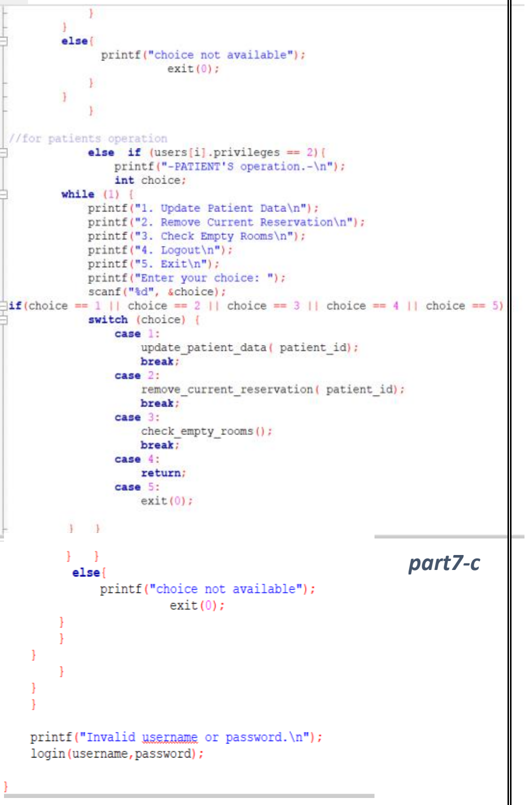
```c
int login(char *username, char *password) {
   int patient_id;
   int privilege;
    printf("Enter your username: ");
    scanf("%s", username);
    printf("Enter your password: ");
    scanf("%s", password);

    for (int i = 0; i < 10; i++){
    if (strcmp(users[i].username, username) == 0) {
        if (strcmp(users[i].password, password) == 0) {
              privilege=users[i].privileges;
          printf("Login successful! Welcome, %s.\n", username);



          // Perform operations based on user privileges
          if (users[i].privileges == 1) {
              printf("-ADMIN'S operations.-\n");
              // Call the functions for an admin
                  char choice;
                  while (1) {
                     printf("1. Update Patient Data\n");
                     printf("2. Remove Current Reservation\n");
                     printf("3. Check Empty Rooms\n");
                     printf("4. Logout\n");
                     printf("5. Exit\n");
                     printf("Enter your choice: ");
                      scanf("%d", &choice);
                      if(choice == 1 || choice == 2 || choice == 3 || choice == 4 || choice == 5){
                      switch (choice) {
              case 1:
                  update_patient_data( patient_id);
                  break;
              case 2:
                  remove_current_reservation( patient_id);
                  break;
```

part7-a



```c
                    case 3:
                        check_empty_rooms();
                        break;
                    case 4:
                        return;
                    case 5:
                        exit(0);


                }}
                     else{
                          printf("choice not available. \n");
                          exit(0);
                         }
                }
                }

//for regular's user operations
                else  if (users[i].privileges == 0){
                    printf("-REGULAR USER'S operation.-\n");
                     int choice;
            while (1) {

                printf("1. Check Empty Rooms\n");
                printf("2. Logout\n");
                 printf("3. Exit\n");
                printf("Enter your choice: ");
                scanf("%d", &choice);
               if(choice == 1 || choice == 2 || choice == 3 ){
                switch (choice) {

                    case 1:
                    check_empty_rooms();
                    break;
                    case 2:
                    return;
                    case 3:
                    exit(0);
```

part7-b



```c
                }
            }
            else{
                  printf("choice not available");
                           exit(0);
                }
            }
                }

//for patients operation
                else  if (users[i].privileges == 2){
                    printf("-PATIENT'S operation.-\n");
                    int choice;
            while (1) {
                printf("1. Update Patient Data\n");
                printf("2. Remove Current Reservation\n");
                printf("3. Check Empty Rooms\n");
                printf("4. Logout\n");
                printf("5. Exit\n");
                printf("Enter your choice: ");
                scanf("%d", &choice);
if(choice == 1 || choice == 2 || choice == 3 || choice == 4 || choice == 5)
                switch (choice) {
                    case 1:
                        update_patient_data( patient_id);
                        break;
                    case 2:
                        remove_current_reservation( patient_id);
                        break;
                    case 3:
                        check_empty_rooms();
                        break;
                    case 4:
                        return;
                    case 5:
                        exit(0);

            }   }

            }   }
             else{
                 printf("choice not available");
                          exit(0);
            }
            }
        }
            }
        }
        }

    printf("Invalid username or password.\n");
    login(username,password);

}
```

part7-c

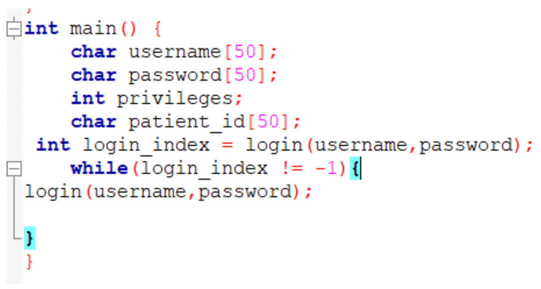
-the function takes in a username and password as input and checks if they match any of the user in the ‘users’ array and access based on the user’s privilege.

-depending on the user’s privilege different options are presented to the user if they user is an admin or patient or regular user by using switch case and calling the previous functions in it again

-If the username or password do not match any user in the array an error message is printed and the function is called again.

- **Lastly the main program:**

- The **main** function initializes some variables and calls the **login** function to authenticate the user. It stores the return value of **login** in the **login_index** variable, which is used to determine whether the user has successfully logged in or not. If **login_index** is not equal to -1, the user has successfully logged in and the **login** function is called again in a loop until the user logs out



```
int main() {
    char username[50];
    char password[50];
    int privileges;
    char patient_id[50];
 int login_index = login(username,password);
    while(login_index != -1){
login(username,password);

}
}
```

- **Conclusion:**
  This code is a simple hospital management system that allows users to log in with a username and password, and perform various operations based on their privileges. The system includes functions to check empty rooms, remove current reservations, and update patient data. The system has an array of users, rooms, and patients, each with their respective structures. The code **scanf** function to take user input and **strcmp** function to compare strings. The **main** function calls the **login** function, which checks if the entered username and password match any of the predefined users. If a match is found, the user is logged in and can perform operations based on their privileges. The **login** function is called again in a loop until the user logs out.

- **Git-hub link:**

https://github.com/sarahalrefaey/sarah