



Department of Software Engineering

Total Marks:

Obtained Marks:

Hospital Patient Queue System

A DSA Semester Project by Group 7

Semester: 4th

Subject: Data Structures & Algorithms

Submitted To: Sir Shakeel Ahmed



Participants

Armughan Naeem

4822-FOC/BSSE/F23

Abdullah Shakeel

4793-FOC/BSSE/F23

Usama Javed

4829-FOC/BSSE/F23

Syed Hussain Ali

4808-FOC/BSSE/F23

Objective:

The objective of this project is to design and implement a hospital patient queue management system in C++ using efficient data structures and algorithms. The system manages patient data, maintains a queue based on priority, handles patient checkups, supports search and sort operations, and improves overall patient flow and data management efficiency.

Introduction:

Hospitals frequently face issues with long queues and inefficient patient handling, which can delay treatment and lower the quality of service. This project provides a software-based solution that prioritizes patients, based on medical urgency using a priority queue. It also allows doctors to manage patients effectively, search patient records, and give their best service.

Data Structures Used:

1. Priority Queue (Max-Heap)

- ⇒ **Implementation:** pQueue class using a vector as the underlying container.
- ⇒ **Purpose:** Ensures highest-priority patients (based on severity 1–10) are served first.
- ⇒ **Operations:** insert(), pop(), top().

2. Hash Table

- ⇒ **Implementation:** hTable class with linear probing (fixed size=35).
- ⇒ **Purpose:** Used for quick patient record lookup using a unique ID.
- ⇒ **Key Mechanism:** Patient ID hashing with collision resolution via linear probing.

3. Vector

- ⇒ **Purpose:** It stores served patients (checked list) for sorting/display.

Searching Algorithms Used:

Linear Search

- ⇒ **Use Case:** It is used to find a patient by ID in the checked patients list.

Binary Search

- ⇒ **Use Case:** Name-based search in the checked list after sorting by name.
- ⇒ **Prerequisite:** Requires selectSort() to pre-sort names.

Hash-Based Search

- ⇒ **Use Case:** Real-time patient lookup in the queue via ID.
- ⇒ **Code:** hTable::search() with linear probing.

Sorting Algorithms Used:

1. Quick Sort

- ⇒ **Use Case:** Sorting checked patients by age (ascending).
- ⇒ **Complexity:** $O(n \log n)$ best & average, $O(n^2)$ worst-case.
- ⇒ **Code:** quickSort() partitioning by age.

2. Selection Sort



- ⇒ **Use Case:** Sorting checked patients by name (ascending).
- ⇒ **Complexity:** $O(n^2)$ all cases.
- ⇒ **Code:** selectSort() iteratively finds lexicographically smallest names.

3. Insertion Sort

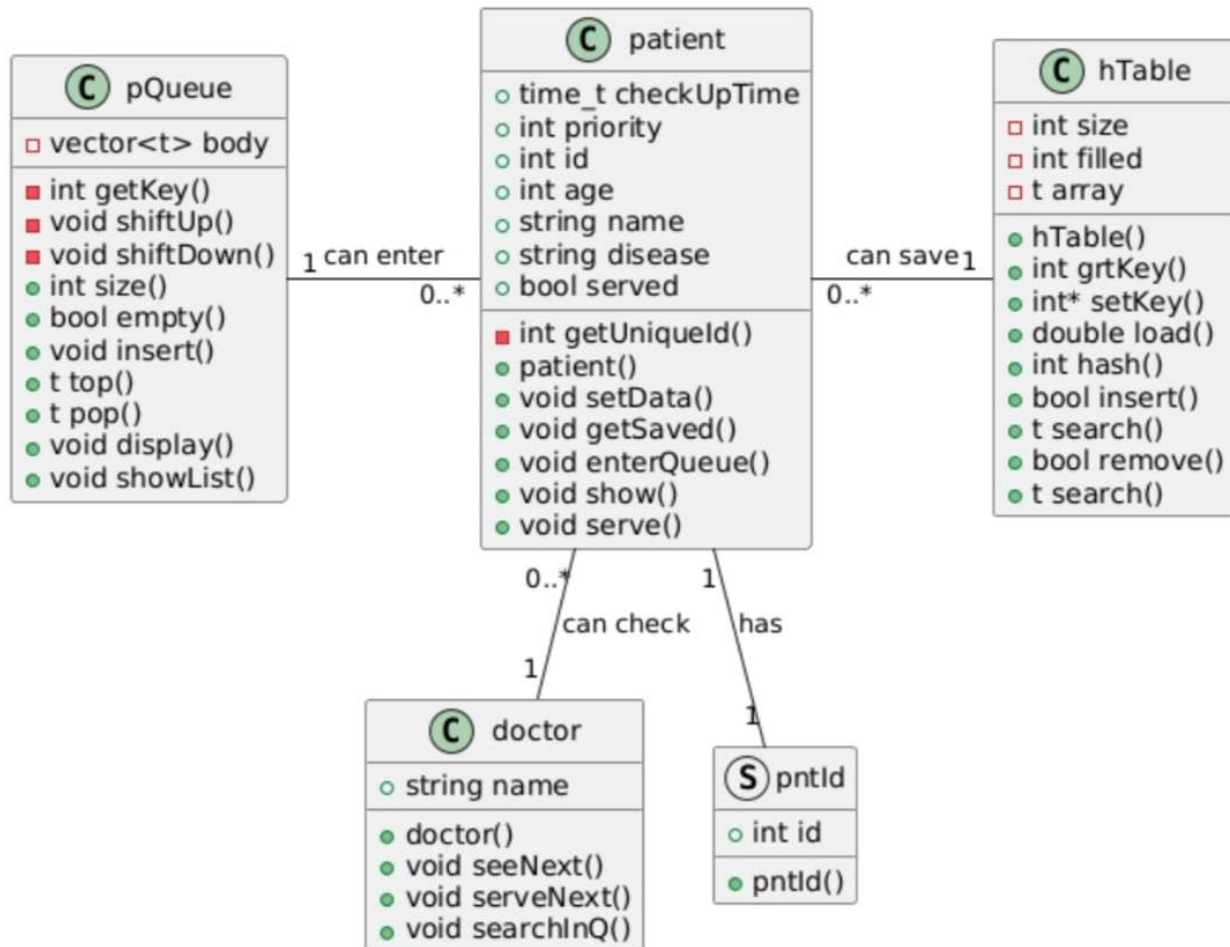
- ⇒ **Use Case:** Sorting checked patients by checkup time (descending).
- ⇒ **Complexity:** $O(n)$ best & $O(n^2)$ average and worst.
- ⇒ **Code:** insertSort() swaps based on chkUpTime.

Core Functionalities:

- ⇒ Add new patient with automated ID generation.
- ⇒ Insert patient into the priority queue.
- ⇒ Display current patient queue.
- ⇒ Doctor sees and serves next patient.
- ⇒ Maintain a checked patient list with checkup timestamp.
- ⇒ Sort and search functionalities on checked list.
- ⇒ Hash table for fast insert, search, and delete by ID.

Class Diagram:

Hospital Patient Queue System



Code:

```
#include<iostream>
#include<cstring>
#include<ctime>
#include<cstdlib>
#include<iomanip>
#include<vector>
#include<memory>

using namespace std;
template <class t>
class pQueue{
    vector<t> body;
    int getKey(t temp){
        return temp.priority;
    }
    void shiftUp(){
        int current=size()-1,parent;
        t temp;
        while(current>0){
            if(!current%2) parent=((current-2)/2);
            else parent=((current-1)/2);
            if(getKey(body[parent])<getKey(body[current])){
                temp=body[parent];
                body[parent]=body[current];
                body[current]=temp;
            }
            current=parent;
        }
    }
    void shiftDown(){
        int current=0,child;
        t temp;
        bool swap=false;
        while(current<size()-1){
            if((current*2+2<size())
                &&
                getKey(body[current*2+1])>getKey(body[current*2+2]))
                child=(current*2+1);
            else if((current*2+2<size())
```

```
        &&
        getKey(body[current*2+2])>getKey(body[current*2+1]))
        child=(current*2+2);
    else if(size()<3) child=current+1;
    if(getKey(body[child])>getKey(body[current])){
        temp=body[current];
        body[current]=body[child];
        body[child]=temp;
        swap=true;
    }
    if(!swap) break;
    swap=false;
    current=child;
}

}
public:
int size(){
    return body.size();
}
bool empty(){
    return size()==0;
}
void insert(t a){
    body.push_back(a);
    shiftUp();
}
t top(){
    t *empty=new t;
    if (size()==0) return *empty;
    delete empty;
    t x=body[0];
    return x;
}
t pop(){
    t *empty=new t;
    if (size()==0) return *empty;
    delete empty;
    t x=body[0];
    body[0]=body[size()-1];
    body.pop_back();
    shiftDown();
    return x;
}
```



```
}
void display(){
    for(int i=0;i<size();i++){
//        cout<<getKey(body[i])
        body[i].show();
        cout<<"\n\n\n";
    }
}
void showList(){
    cout<<left<<setw(5)<<"num"
        <<setw(7)<<"Id"
        <<setw(15)<<"Name"
        <<setw(10)<<"priority"
        <<setw(5)<<"Age"
        <<setw(15)<<"Disease"
        <<endl<<endl;
    for(int i=0;i<size();i++){
        cout<<left<<setw(5)<<i+1
            <<setw(7)<<body[i].id
            <<setw(15)<<body[i].name
            <<setw(10)<<body[i].priority
            <<setw(5)<<body[i].age
            <<setw(15)<<body[i].disease
            <<endl;
    }
}
};
template <class t>
class hTable{
    int size=35,filled=0;
//    unique_ptr<t[]> array(new t[size]);
//    t *array=new t[size];;
    t array[35];

    void start(){
//        for (int i=0;i<size;i++)
//            array[i]= new t;
    }
public:
    hTable(){}
//    hTable(int a){
//        size = a;
```

```
//      start();
//      }
~hTable(){
//      delete []array;
}
int getKey(t temp){
    return temp.id;
}
int* setKey(t &temp){
    return &temp.id;
}
double load(){
    return (static_cast<double>(filled)/static_cast<double>(size));
}
int hash(int a){
    return a%size;
}
bool insert(t realKey){
    int key=getKey(realKey);
    for (int i=0;i<size;i++){
        if(getKey(array[hash(key)])==-1){
            array[hash(key)]=realKey;
            filled++;
            return true;
        }
        key = key+i;
    }
    return false;
}
t search(int realKey){
    int key=realKey;
    for (int i=0;i<size;i++){
        if(getKey(array[hash(key)])==realKey){
            return array[hash(key)];
        }
        key = key+i;
    }
    t *empty =new t;
    return *empty;
}
bool remove(t &realKey){
    int key=getKey(realKey);
```

```
for (int i=0;i<size;i++){
    if(getKey(array[hash(key)])==getKey(realKey)){
        int *num=setKey(array[hash(key)]);
        *num=-1;
        filled--;
        return true;
    }
    key = key+i;
}
return false;
}

void display(){
    for (int i=0;i<size;i++){
        if(getKey(array[i])!=-1){
            cout<<"value at index "
                <<i<<" is "<<getKey(array[i])/*show()*/
                <<"\n";
        }
    }
}

};

struct pntId{
    int id;
    pntId (){
        id = -1;
    }
    pntId (int a){
        id = a;
    }
};

class patient{
    int getUniqueId(hTable <pntId> &t){
        int a=(1000+(rand()%(10000-1000)));
        if(!(t.search(a)).id== -1){
            a = 1000+(rand()%(10000-1000));
        }
        pntId *b= new pntId(a);
        pntId p=*b;
        t.insert(p);
        return p.id;
    }
public:
```

```
time_t chkUpTime;
int priority,id,age;
string name,disease;
bool served;
patient(){
    priority = -1;
    id=-1;
    age=-1;
    name="unknown";
    disease="unknown";
    served=false;
}
patient(int a,int c,string d,string e,hTable<pntId> &IDs){
    priority = a;
    id=getUniqueId(IDs);
    age=c;
    name=d;
    disease=e;
    served=false;
}
void setData(hTable<pntId> &IDs){
    id=getUniqueId(IDs);
    cout<<"\n generated id: "<<id;
    cout<<"\n enter name: ";
    cin>>name;
    cout<<"enter age: ";
    cin>>age;
    while(!(age>=1&&age<=149)){
        cout<<"\n invalid INPUT. \n enter again";
        cin>>age;
    }

    cout<<"enter disease: ";
    cin>>disease;
    cout<<"enter priority:(1 to 10) ";
    cin>>priority;
    while(!(priority>=1&&priority<=10)){
        cout<<"\n invalid INPUT. \n enter again";
        cin>>priority;
    }
}
void getSaved(hTable<patient> &t){
    if(t.insert(*this)) cout<<"entered table";
```

```
        else cout<<"can't enter table";
    }
    void enterQueue(pQueue<patient>& q,hTable<patient> &t){
        q.insert(*this);
        t.insert(*this);
    }
    void show(){
        cout<<"\nname: "<<name;
        cout<<"\nage: "<<age;
        cout<<"\nid: "<<id;
        cout<<"\npriority: "<<priority;
        cout<<"\nserved :"<<served;
        if(served)
            cout<<"\nserved at: "<<put_time(localtime(&chkUpTime),"%Y-%m-%d-
%H:%M:%S");
        cout<<"\n";
    }
    void serve(){
        chkUpTime=time(0);
        served=true;
    }
};
class doctor{
public:
    string name;
    doctor(string n){
        name= n;
    }
    void seeNext(pQueue<patient> &q){
        patient p=q.top();
        p.show();
    }
    void serveNext(pQueue<patient> &q,hTable<patient> &t1,vector<patient>
&chkd){
        patient p=q.pop();
        t1.remove(p);
        p.serve();
        chkd.push_back(p);
    }
    void searchInQ(hTable<patient> &t){
        patient p;
        int a;
```

```
        cout<<"\nenter id to search:";
        cin>>a;
        while(!(a>=1000&&a<=9999)){
            cout<<"\ninvalid id. \n enter again";
            cin>>a;
        }
        p=t.search(a);
        if(p.id==-1) cout <<"\nnot found\n";
        else {
            cout <<"\nfound in queue\n";
            p.show();
        }
    }
};

int qSort(vector<patient> &a,int min,int max){
    int pos=min;
    patient temp;
    for (int i=min;i<=max;i++){
        if(a[i].age<=a[max].age){
            temp=a[i];
            a[i]=a[pos];
            a[pos]=temp;
            pos++;
        }
    }
    return pos-1;
}

void quickSort(vector<patient> &a,int min,int max){
    if(min>=max)
        return;
    int pivot = qSort(a,min,max);
    quickSort(a,min,pivot-1);
    quickSort(a,pivot+1,max);
}

void insertSort(vector<patient> &a,int max){
    patient temp;
    for (int i=0;i<max;i++){
        for(int j=i;j>0;j--){
            if(a[j].chkUpTime>a[j-1].chkUpTime){
                temp=a[j-1];
                a[j-1]=a[j];
                a[j]=temp;
            }
        }
    }
}
```

```
    }
    else
        break;
    }
}

void selectSort(vector<patient> &a,int max){
    patient min,temp;
    int loc;
    for (int i=0;i<max;i++){

        min=a[i];
        loc=i;

        for(int j=i+1;j<max;j++){
            if(a[j].name<min.name){
                min=a[j];
                loc=j;
            }
        }
        temp=a[i];
        a[i]=a[loc];
        a[loc]=temp;
    }
}

int binarySearch(vector<patient> &a,string val){
    int size=a.size(), low=0, high=size-1 ,mid;

    while(low<=high){
        mid=low+(high-low)/2;
        if(a[mid].name==val)
            return mid;
        else if(a[mid].name<val)
            low = mid+1;
        else
            high=mid-1;
    }
    return -1;
}

int main(){
    cout<<"Bismillah hirrehman nirraheem\n";
```

```
srand(static_cast<int>(time(0)));
hTable<pntId> IDs;
hTable<patient> inQueue;
pQueue<patient> q1;
doctor dctr("Dr.Kamal");
vector<patient> checked;
int choice,choice2,b,c;
char s;

patient p1(7,15,"ali","fever",IDs);
patient p2(10,23,"bilal","injury",IDs);
patient p3(5,17,"saad","cough",IDs);
patient p4(3,19,"khalid","itching",IDs);
patient p5(1,14,"jaffer","flu",IDs);
patient p6(7,15,"karam","fever",IDs);
patient p7(10,28,"raza","injury",IDs);
patient p8(5,27,"hashim","cough",IDs);
patient p9(3,29,"khurshid","itching",IDs);
patient p10(1,5,"zia","flu",IDs);
patient p11(7,11,"wasee","fever",IDs);
patient p12(10,23,"umer","injury",IDs);
patient p13(5,7,"toqeer","cough",IDs);
patient p14(3,9,"nadir","itching",IDs);
patient p15(1,16,"fahad","flu",IDs);
patient p16(1,5,"yasir","flu",IDs);
patient p17(7,11,"wajih","fever",IDs);
patient p18(10,23,"qarir","injury",IDs);
patient p19(5,7,"qamar","cough",IDs);
patient p20(3,9,"nasir","itching",IDs);
patient p21(1,16,"javed","flu",IDs);

time_t smplTime=time(0);

p1.chkUpTime=smplTime-2500;
p2.chkUpTime=smplTime-4200;
p3.chkUpTime=smplTime-6900;
p4.chkUpTime=smplTime-8600;
p5.chkUpTime=smplTime-1000;
p6.chkUpTime=smplTime-1700;
p7.chkUpTime=smplTime-3300;
p8.chkUpTime=smplTime-2700;
p9.chkUpTime=smplTime-3900;
```




```
p10.chkUpTime=smplTime-1500;
p11.chkUpTime=smplTime-5300;
p12.chkUpTime=smplTime-3700;
p13.chkUpTime=smplTime-4900;

checked.push_back(p1);
checked.push_back(p2);
checked.push_back(p3);
checked.push_back(p4);
checked.push_back(p5);
checked.push_back(p6);
checked.push_back(p7);
checked.push_back(p8);
checked.push_back(p9);
checked.push_back(p10);
checked.push_back(p11);
checked.push_back(p12);
checked.push_back(p13);

p14.enterQueue(q1,inQueue);
p15.enterQueue(q1,inQueue);
p16.enterQueue(q1,inQueue);
p17.enterQueue(q1,inQueue);
p18.enterQueue(q1,inQueue);
p19.enterQueue(q1,inQueue);
p20.enterQueue(q1,inQueue);
p21.enterQueue(q1,inQueue);

do{
    system("cls");
    cout<<"Bismillah hirrehman nirraheem\n";
    cout<<"\n PATIENT QUEUE MANAGEMENT  \n\n";
    cout<<"\nEnter 1 to add new new patient"
        <<"\nEnter 2 to check next patient"
        <<"\nEnter 3 to find in queue"
        <<"\nEnter 4 to view patients of queue"
        <<"\nEnter 5 to view doctor name"
        <<"\nEnter 6 to view checked list"
        <<"\nEnter 7 to sort checked list"
        <<"\nEnter 8 to search by id"
        <<"\nEnter 9 to search by name"
        <<" \n your choice : ";
```



```
cin>>choice;
while(!(choice>=1&&choice<=9)){
    cout<<"\ninvalid choice. \n enter again";
    cin>>choice;
}
switch(choice) {
    case 1 :{
        system("cls");
        cout<<"NEW PATIENT\n";
        patient *a=new patient;
        patient p=*a;
        p.setData(IDs);
        p.enterQueue(q1,inQueue);
        break;
    }
    case 2 :{
        system("cls");
        cout<<"CHECKUP ROOM\n";
        cout<<"\npatient being checked :\n";
        dctr.seeNext(q1);
        cout<<"\npatient got checked ?(y)";
        cin>>s;
        while(!(s=='y' || s=='Y')){
            cout<<"\ninvalid input. \n enter y or Y";
            cin>>s;
        }
        dctr.serveNext(q1,inQueue,checked);
        break;
    }
    case 3 :{
        system("cls");
        cout<<"QUEUE SEARCH\n";
        dctr.searchInQ(inQueue);
        break;
    }
    case 4 :{
        system("cls");
        cout<<"QUEUE LIST\n";
        q1.showList();
        break;
    }
    case 5 :{
```



```
system("cls");
cout<<"THE DOCTOR\n";
cout<<"\n your doctoer's name is : "<<ctr.name;
break;
}
case 6 :{
system("cls");
cout<<"CHECKED PATIENTS LIST\n";
cout<<left<<setw(5)<<"num"
    <<setw(7)<<"Id"
    <<setw(15)<<"Name"
    <<setw(5)<<"Age"
    <<setw(15)<<"Disease"
    <<setw(35)<<"Date & Time of checkup"
    <<endl
    <<endl;
for(int i=0;i<checked.size();i++){
cout<<left<<setw(5)<<i+1
    <<setw(7)<<checked[i].id
    <<setw(15)<<checked[i].name
    <<setw(5)<<checked[i].age
    <<setw(15)<<checked[i].disease
    <<setw(35)<<put_time(localtime(&checked[i].chkUpTime),"%Y-%m-%d %H:%M:%S")
    <<endl;
}
break;
}
case 7 :{
system("cls");
cout<<"SORTING\n";
cout<<"\nEnter 1 to sort by age"
    <<"\nEnter 2 to sort by name"
    <<"\nEnter 3 to sort by time"
    <<" \n your choice : ";
cin>>choice2;
while(!(choice2>=1&&choice2<=3)){
cout<<"\ninvalid choice. \n enter again";
cin>>choice2;
}
switch(choice2) {
case 1:{
```

```
        quickSort(checked,0,checked.size()-1);
        break;
    }
    case 2:{
        selectSort(checked,checked.size());
        break;
    }
    case 3:{
        insertSort(checked,checked.size());
        break;
    }
    default :{
        break;
    }
}
cout<<"sorted";
break;
}
case 8 :{
    system("cls");
    cout<<"ID SEARCH\n";
    int tmpId;
    bool found=false;
    cout<<"\nenter Id :";
    cin>>tmpId;
    while(!(tmpId>=1000&&tmpId<=9999)){
        cout<<"\ninvalid id. \n enter again";
        cin>>tmpId;
    }
    for(int i=0;i<checked.size();i++){
        if(tmpId==checked[i].id){

            checked[i].show();
            found=true;
        }
    }
    if(!found)
        cout<<"\nnot found";
    break;
}
case 9 :{
    system("cls");
```



```
        cout<<"NAME SEARCH\n";
        string tmpname;
        int index;
        cout<<"\nenter name :";
        cin>>tmpname;
        selectSort(checked,checked.size());
        index=binarySearch(checked,tmpname);
        if(index==-1)cout<<"\nnot found";
        else checked[index].show();
        break;
    }
    default:
        break;
}
cout<<"\nenter 0 to quit and 1 to go back: ";
cin>>c;
}while (!c==0);
return 0;
}
```



Implementation Overview:

1. Initialization:

- ⇒ hTable <ptnId> ensures unique patient IDs.
- ⇒ Sample patients pre-loaded into pQueue and checked for testing.

2. Main Menu:

- ⇒ 9 interactive options (add patient, serve next, search, display, sort, etc.).

3. Patient Flow:

- ⇒ New patients: Added to pQueue and hTable.
- ⇒ Served patients: Moved from pQueue to checked vector with timestamp.

4. Data Query:

- ⇒ Sorting: Triggered via quickSort(), selectSort(), or insertSort().
- ⇒ Searching: Hash table (active queue) or linear/binary search (checked list).

Screenshots of System's Working:

Main Menu:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Bismillah hirrehman nirraheem

PATIENT QUEUE MANAGEMENT

Enter 1 to add new new patient
Enter 2 to check next patient
Enter 3 to find in queue
Enter 4 to view patients of queue
Enter 5 to view doctor name
Enter 6 to view checked list
Enter 7 to sort checked list
Enter 8 to search by id
Enter 9 to search by name
your choice : █
```

Initial Queue List:

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

QUEUE LIST
num  Id      Name      priority  Age  Disease
1    1299    qarir      10        23   injury
2    1644    wajih      7         11   fever
3    4098    qamar      5         7    cough
4    6418    fahad      1         16   flu
5    7729    nadir      3         9    itching
6    4521    yasir      1         5    flu
7    5919    nasir      3         9    itching
8    8868    javed      1         16   flu

enter 0 to quit and 1 to go back: █
```



Initial Checked Patients list:

PROBLEMS	2	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
CHECKED PATIENTS LIST					
num	Id	Name	Age	Disease	Date & Time of checkup
1	7232	ali	15	fever	2025-05-31 12:43:31
2	6978	bilal	23	injury	2025-05-31 12:15:11
3	9588	saad	17	cough	2025-05-31 11:30:11
4	3756	khalid	19	itching	2025-05-31 11:01:51
5	6249	jaffer	14	flu	2025-05-31 13:08:31
6	5241	karam	15	fever	2025-05-31 12:56:51
7	6377	raza	28	injury	2025-05-31 12:30:11
8	1521	hashim	27	cough	2025-05-31 12:40:11
9	5991	khurshid	29	itching	2025-05-31 12:20:11
10	3422	zia	5	flu	2025-05-31 13:00:11
11	1692	wasee	11	fever	2025-05-31 11:56:51
12	3786	umer	23	injury	2025-05-31 12:23:31
13	6594	toqeer	7	cough	2025-05-31 12:03:31
enter 0 to quit and 1 to go back:					



Adding New Patient

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

NEW PATIENT

generated id: 1568
enter name: Hameed
enter age: 19
enter disease: Cough
enter priority:(1 to 10) 4

enter 0 to quit and 1 to go back: █
```

Queue list After adding New Patient:

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

QUEUE LIST
num  Id      Name      priority  Age  Disease
1    1299    qarir      10        23   injury
2    1644    wajih      7         11   fever
3    4098    qamar      5         7    cough
4    1568    Hameed     4         19   Cough
5    7729    nadir      3         9    itching
6    4521    yasir      1         5    flu
7    5919    nasir      3         9    itching
8    8868    javed      1        16   flu
9    6418    fahad      1        16   flu

enter 0 to quit and 1 to go back: █
```



Checking a Patient:

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

CHECKUP ROOM

patient being checked :

name: qarir
age: 23
id: 1299
priority: 10
served :0

patient got checked ?(y)y

enter 0 to quit and 1 to go back: █
```

Queue list After Checking one Patient:

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

QUEUE LIST
num  Id    Name      priority  Age  Disease
1    1644  wajih     7         11   fever
2    1568  Hameed    4         19   Cough
3    4098  qamar     5         7    cough
4    6418  fahad     1         16   flu
5    7729  nadir     3         9    itching
6    4521  yasir     1         5    flu
7    5919  nasir     3         9    itching
8    8868  javed     1         16   flu

enter 0 to quit and 1 to go back: █
```



Checked Patients list After checking one Patient:

PROBLEMS	2	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
CHECKED PATIENTS LIST					
num	Id	Name	Age	Disease	Date & Time of checkup
1	7232	ali	15	fever	2025-05-31 12:43:31
2	6978	bilal	23	injury	2025-05-31 12:15:11
3	9588	saad	17	cough	2025-05-31 11:30:11
4	3756	khalid	19	itching	2025-05-31 11:01:51
5	6249	jaffer	14	flu	2025-05-31 13:08:31
6	5241	karam	15	fever	2025-05-31 12:56:51
7	6377	raza	28	injury	2025-05-31 12:30:11
8	1521	hashim	27	cough	2025-05-31 12:40:11
9	5991	khurshid	29	itching	2025-05-31 12:20:11
10	3422	zia	5	flu	2025-05-31 13:00:11
11	1692	wasee	11	fever	2025-05-31 11:56:51
12	3786	umer	23	injury	2025-05-31 12:23:31
13	6594	toqeer	7	cough	2025-05-31 12:03:31
14	1299	qarir	23	injury	2025-05-31 13:33:27
enter 0 to quit and 1 to go back: <input type="text"/>					



Find/Search a Patient in Queue:

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

QUEUE SEARCH

enter id to search:4521

found in queue

name: yasir
age: 5
id: 4521
priority: 1
served :0

enter 0 to quit and 1 to go back: |
```

View Doctor's name:

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

THE DOCTOR

your doctoer's name is :Dr.Kamal
enter 0 to quit and 1 to go back: |
```



Sort Checked Patients:

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

SORTING

Enter 1 to sort by age
Enter 2 to sort by name
Enter 3 to sort by time
your choice : █
```

Sort Checked Patients by Age:

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

SORTING

Enter 1 to sort by age
Enter 2 to sort by name
Enter 3 to sort by time
your choice : 1
sorted
enter 0 to quit and 1 to go back: █
```



Checked Patients List, after Sorting Checked Patients by Age:

PROBLEMS	2	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
CHECKED PATIENTS LIST					
num	Id	Name	Age	Disease	Date & Time of checkup
1	3422	zia	5	flu	2025-05-31 13:00:11
2	6594	toqeer	7	cough	2025-05-31 12:03:31
3	1692	wasee	11	fever	2025-05-31 11:56:51
4	6249	jaffer	14	flu	2025-05-31 13:08:31
5	5241	karam	15	fever	2025-05-31 12:56:51
6	7232	ali	15	fever	2025-05-31 12:43:31
7	9588	saad	17	cough	2025-05-31 11:30:11
8	3756	khalid	19	itching	2025-05-31 11:01:51
9	3786	umer	23	injury	2025-05-31 12:23:31
10	6978	bilal	23	injury	2025-05-31 12:15:11
11	1299	qarir	23	injury	2025-05-31 13:33:27
12	1521	hashim	27	cough	2025-05-31 12:40:11
13	6377	raza	28	injury	2025-05-31 12:30:11
14	5991	khurshid	29	itching	2025-05-31 12:20:11
enter 0 to quit and 1 to go back: █					



Sort Checked Patients by Name:

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

SORTING

Enter 1 to sort by age
Enter 2 to sort by name
Enter 3 to sort by time
your choice : 2
sorted
enter 0 to quit and 1 to go back: █
```

Checked Patients List, after Sorting Checked Patients by Name:

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

CHECKED PATIENTS LIST
num  Id    Name      Age  Disease      Date & Time of checkup
1    7232    ali       15   fever        2025-05-31  12:43:31
2    6978    bilal     23   injury       2025-05-31  12:15:11
3    1521    hashim    27   cough        2025-05-31  12:40:11
4    6249    jaffer    14   flu          2025-05-31  13:08:31
5    5241    karam     15   fever        2025-05-31  12:56:51
6    3756    khalid    19   itching      2025-05-31  11:01:51
7    5991    khurshid  29   itching      2025-05-31  12:20:11
8    1299    qarir     23   injury       2025-05-31  13:33:27
9    6377    raza      28   injury       2025-05-31  12:30:11
10   9588    saad      17   cough        2025-05-31  11:30:11
11   6594    toqeer    7    cough        2025-05-31  12:03:31
12   3786    umer      23   injury       2025-05-31  12:23:31
13   1692    wasee     11   fever        2025-05-31  11:56:51
14   3422    zia       5    flu          2025-05-31  13:00:11

enter 0 to quit and 1 to go back: █
```

Sort Checked Patients by Checkup Time:

```

PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

SORTING

Enter 1 to sort by age
Enter 2 to sort by name
Enter 3 to sort by time
your choice : 3
sorted
enter 0 to quit and 1 to go back: █

```

Checked Patients List after Sorting Checked Patients by Checkup Time:

```

PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

CHECKED PATIENTS LIST
num Id      Name      Age  Disease      Date & Time of checkup
1   1299  qarir    23   injury    2025-05-31  13:33:27
2   6249  jaffer   14   flu       2025-05-31  13:08:31
3   3422  zia      5    flu       2025-05-31  13:00:11
4   5241  karam    15   fever    2025-05-31  12:56:51
5   7232  ali      15   fever    2025-05-31  12:43:31
6   1521  hashim   27   cough    2025-05-31  12:40:11
7   6377  raza     28   injury    2025-05-31  12:30:11
8   3786  umer     23   injury    2025-05-31  12:23:31
9   5991  khurshid 29   itching   2025-05-31  12:20:11
10  6978  bilal    23   injury    2025-05-31  12:15:11
11  6594  toqeer   7    cough    2025-05-31  12:03:31
12  1692  wasee    11   fever    2025-05-31  11:56:51
13  9588  saad     17   cough    2025-05-31  11:30:11
14  3756  khalid   19   itching   2025-05-31  11:01:51

enter 0 to quit and 1 to go back: █

```




Search Checked Patients by ID:

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

ID SEARCH

enter Id :5991

name: khurshid
age: 29
id: 5991
priority: 3
served :0

enter 0 to quit and 1 to go back: █
```

Search Checked Patients by Name:

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

NAME SEARCH

enter name :umer

name: umer
age: 23
id: 3786
priority: 10
served :0

enter 0 to quit and 1 to go back: █
```



Performance Analysis:

Component	Operation	Time Complexity	Remarks
Priority Queue	insert()	$O(\log n)$	Heapify via shiftUp()
	pop()	$O(\log n)$	Heapify via shiftDown()
Hash Table	search()	$O(1)$ best, $O(n)$ worst	Linear probing collisions
	insert()	$O(1)$ best, $O(n)$ worst	Load factor = 35 fixed
Quick Sort	Sort by age	$O(n \log n)$ avg	Efficient for large datasets
Selection Sort	Sort by name	$O(n^2)$	Inefficient for large n
Insertion Sort	Sort by Checkup time	$O(n^2)$	Inefficient for large n
Binary Search	Search by name	$O(\log n)$	Requires pre-sorting ($O(n^2)$)
Linear Search	Search by ID (Checked Patients)	$O(n)$	Inefficient for very large n

Member-wise Contribution:

1. Armughan Naeem (4822-FOC/BSSE/F23)

Core Data Structure Implementation:

- ⇒ Designed and implemented the priority queue (pQueue class) using a max-heap backed by a vector.
- ⇒ Developed critical heap operations:
 - shiftUp(): Ensures heap property after insertion ($O(\log n)$).
 - shiftDown(): Maintains heap property after extraction ($O(\log n)$).
- ⇒ Created queue management functions: insert(), pop(), top(), and showList() for patient display.

System Integration:

- ⇒ Integrated the priority queue with the patient management workflow (enterQueue()).
- ⇒ Implemented heap-based prioritization logic to ensure critical patients are served first.

2. Abdullah Shakeel (4793-FOC/BSSE/F23)

Hash Table & ID System

- ⇒ Make the hash table (hTable class) with linear probing for collision resolution.
- ⇒ Implemented key hashing operations: insert(), search(), and remove() ($O(1)$ avg).
- ⇒ Designed the unique patient ID generation system:
 - Auto-generated 4-digit IDs (1000–9999) using rand().
 - Ensured uniqueness via hash table validation (getUniqueId()).

Patient Registration

- ⇒ Developed patient data capture logic (setData()) with input validation.

3. Usama Javed (4829-FOC/BSSE/F23)

Sorting Algorithms & UI

- ⇒ Implemented all sorting algorithms for Sorting the patients according to the needs:
 - **Quick Sort:** Age-based sorting (quickSort() and qSort()).
 - **Selection Sort:** Lexicographical name sorting (selectSort()).
 - **Insertion Sort:** Checkup time-based ordering (insertSort()).
- ⇒ Built the console UI system:
 - Designed the interactive menu flow in main().
 - Created data display logic for queues/checked patients.

Doctor Operations

- Coded doctor workflows: seeNext(), serveNext(), and queue integration.

4. Syed Hussain Ali (4808-FOC/BSSE/F23)

Searching Algorithms & Debugging

- ⇒ Implemented search systems:
 - **Hash-based search:** Real-time patient lookup in active queue (searchInQ()).
 - **Binary Search:** Name-based search in checked list (binarySearch()), requiring pre-sorting.



- **Linear Search:** ID-based scan in checked patients.

Testing & Optimization

- ⇒ Tested everything and fixed bugs.
- ⇒ Handle the edge-cases.

Summary:

This project effectively demonstrates the real-world use of data structures and algorithms in managing critical systems like hospital patient queues. It uses object-oriented principles and modular code to ensure maintainability and clarity. Key operations such as insertion, deletion, searching, and sorting are implemented efficiently using standard algorithms.

Future Improvements:

- I. File I/O to save patient data between runs.
- II. GUI for user-friendly interactions.
- III. Time-complexity optimization for sorting large datasets.
- IV. Expand system to handle multiple doctors and departments.



Conclusion:

This project demonstrates a practical application of DSA concepts to solve real-world inefficiencies in healthcare management. It enhances hospital efficiency by reducing patient waiting time and supporting medical staff in managing patient flow. By applying priority queues, hash tables, and sorting/searching algorithms, the system ensures critical patients receive timely care while maintaining efficient data tracking. With additional features and UI improvements, it can evolve into a complete hospital management solution.

Links:

<https://github.com/786Abdullah/HospitalPatientQueueSystem>

<https://www.linkedin.com/in/abdullah-shakeel-743b48368?>

