**HOSPITAL MANAGEMENT SYSTEM**

**INTRODUCTION**

Hospital are the essential part of our lives, providing best medical facilities to people suffering from various heart diseases, lungs diseases and someone want to do plastic surgery etc. It is necessary for the hospitals to keep track of its day-to-day activities & records of its patients for heart clinic, lungs clinic, plastic surgery that keep the hospital running smoothly & successfully.

But keeping track of all the activities and their records on paper is very cumbersome and error prone. It also is very inefficient and a time-consuming process Observing the continuous increase in population and number of people visiting the hospital. Recording and maintaining all these records is highly unreliable, inefficient and error-prone. It is also not economically & technically feasible to maintain these records on paper.

Thus, keeping the working of the manual system as the basis of our project. We have developed an automated version of the manual system, named as “Hospital Management System”. The main aim of our project is to provide a paper-less hospital up to 90%. It also aims at providing low-cost reliable automation of the existing systems. The system also provides excellent security of data at every level of user-system interaction and also provides robust & reliable storage and backup facilities. Hospital Management System (HMS) is a computer system that facilitates managing the functioning of the hospital or any medical set up. This system or software will help in making the whole functioning paperless. It integrates all the information regarding patients’ details, list of patients etc. into one software.

**It uses the concept of following C++ topics**:

* Loops
* Functions
* If Else
* Switch
* Classes

**SOFTWARE AND HARDWARE REQUIREMENTS**

**Software Requirements:** -

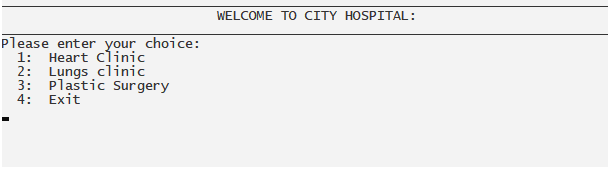
* Operating System: Windows XP
* Programming Language: C++
* Software Used: CodeBlocks, MS Word and Notepad++

**Hardware Requirements:** -

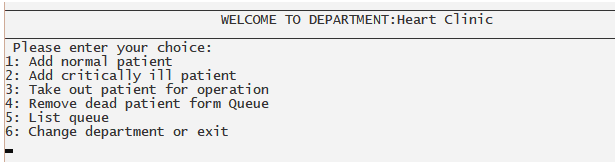
* Pentium-IV (Processor)
* 256 MB Ram
* 512 KB Cache Memory
* Hard Disk 10 GB

**SCREEN SHOT**

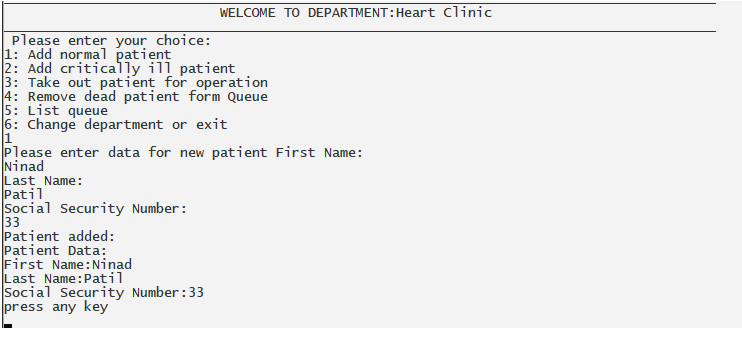
**MAIN MENU**



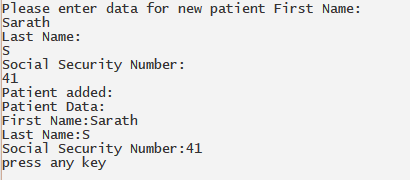
**HEART CLINIC**



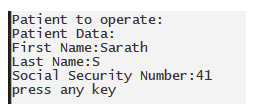
**ADD NORMAL PATIENT**

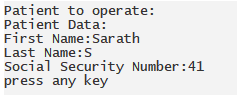


**ADD CRITICALLY ILL PATIENT**

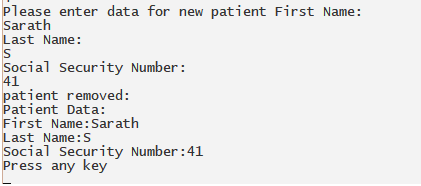


**TAKE OUT PATIENT FOR OPERATION**

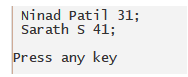




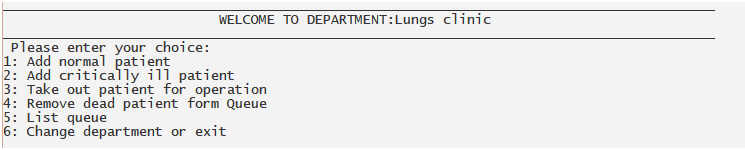
**REMOVE DEAD PATIENT FROM QUEUE**



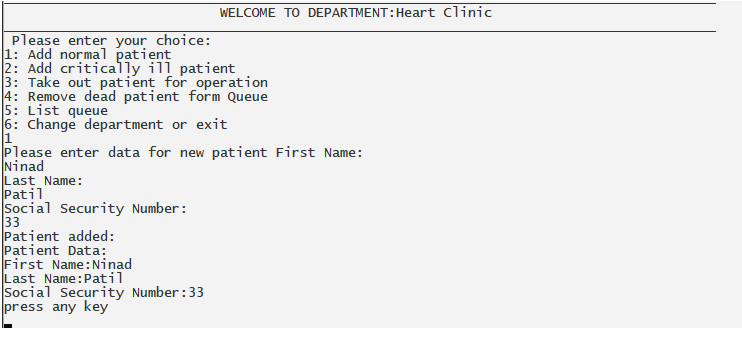
**LIST QUEUE**



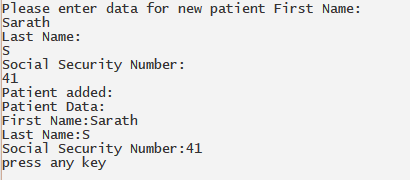
**LUNGS CLINIC**



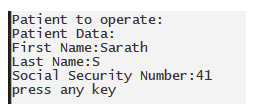
**ADD NORMAL PATIENT**

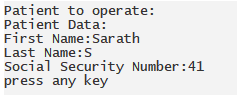


**ADD CRITICALLY ILL PATIENT**

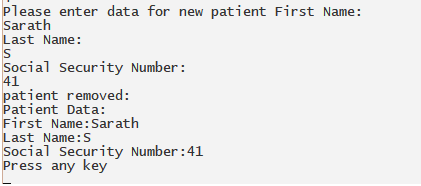


**TAKE OUT PATIENT FOR OPERATION**

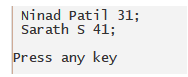




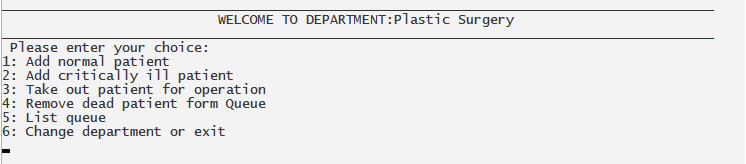
**REMOVE DEAD PATIENT FROM QUEUE**



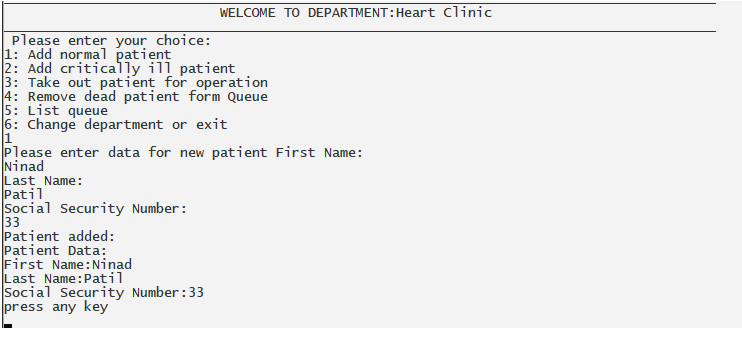
**LIST QUEUE**



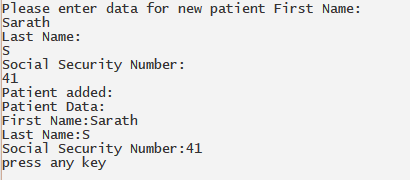
**PLASTIC SURGERY**



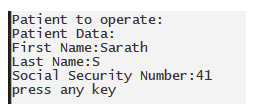
**ADD NORMAL PATIENT**

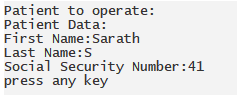


**ADD CRITICALLY ILL PATIENT**

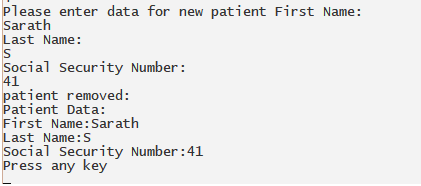


**TAKE OUT PATIENT FOR OPERATION**

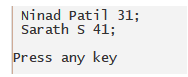




**REMOVE DEAD PATIENT FROM QUEUE**



**LIST QUEUE**



**SOURCE CODE**

#include<iostream>

#include<conio.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define MAXPATIENTS 100

using namespace std;

struct patient

{

char FirstName[20];

char LastName[20];

char ID[15];

};

class queue

{

public:

queue(void);

int AddPatientAtEnd(patient p);

int AddPatientAtBeginning(patient p);

patient GetNextPatient(void);

int RemoveDeadPatient(patient \*p);

void OutputList(void);

char DepartmentName[50];

private:

int NumberOfPatients;

patient List[MAXPATIENTS];

};

queue::queue()

{

NumberOfPatients=0;

}

int queue::AddPatientAtEnd(patient p)

{

if(NumberOfPatients>=MAXPATIENTS)

{

return 0;

}

else

List[NumberOfPatients]=p;

NumberOfPatients++;

return 1;

}

int queue::AddPatientAtBeginning(patient p)

{

int i;

if(NumberOfPatients>=MAXPATIENTS)

{

return 0;

}

for(i=NumberOfPatients-1;i>=0;i--)

{

List[i+1]=List[i];

}

List[0]=p;

NumberOfPatients++;

return 1;

}

patient queue::GetNextPatient(void)

{

int i;

patient p;

if(NumberOfPatients==0)

{

strcpy(p.ID,"");

return p;

}

p=List[0];

NumberOfPatients--;

for(i=0;i<NumberOfPatients;i++)

{

List[i]=List[i+1];

}

return p;

}

int queue::RemoveDeadPatient(patient \*p)

{

int i,j,found=0;

for(i=0;i<NumberOfPatients;i++)

{

if(stricmp(List[i].ID,p->ID)==0)

{

\*p=List[i];

found=1;

NumberOfPatients;

for(j=1;j<NumberOfPatients;j++)

{

List[j]=List[j+1];

}

}

}

return found;

}

void queue::OutputList(void)

{

int i;

if(NumberOfPatients==0)

{

cout<<"Queue is empty"<<endl;

}

else

{

for(i=0;i<NumberOfPatients;i++)

{

cout<<" "<<List[i].FirstName;

cout<<" "<<List[i].LastName;

cout<<" "<<List[i].ID;

cout<<";"<<endl;

}

}

}

patient InputPatient(void)

{

patient p;

cout<<"Please enter data for new patient First Name:"<<endl;

cin.getline(p.FirstName,sizeof(p.FirstName));

cout<<"Last Name:"<<endl;

cin.getline(p.LastName,sizeof(p.LastName));

cout<<"Social Security Number:"<<endl;

cin.getline(p.ID,sizeof(p.ID));

if(p.FirstName[0]==0||p.LastName[0]==0||p.ID[0]==0)

{

strcpy(p.ID,"");

cout<<"Error:Data not valid,operation cancelled."<<endl;

getch();

}

return p;

}

void OutputPatient(patient \*p)

{

if (p==NULL||p->ID[0]==0)

{

cout<<"No Patient"<<endl;

return;

}

else

cout<<"Patient Data:"<<endl;

cout<<"First Name:"<<p->FirstName<<endl;

cout<<"Last Name:"<<p->LastName<<endl;

cout<<"Social Security Number:"<<p->ID<<endl;

}

int ReadNumber()

{

char buffer[20];

cin.getline(buffer,sizeof(buffer));

return atoi(buffer);

}

void DepartmentMenu(queue \*q)

{

int choice=0,success;

patient p;

while(choice !=6)

{

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"\t"" WELCOME TO DEPARTMENT:"<<q->DepartmentName<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<" Please enter your choice:"<<endl;

cout<<"1: Add normal patient"<<endl;

cout<<"2: Add critically ill patient"<<endl;

cout<<"3: Take out patient for operation"<<endl;

cout<<"4: Remove dead patient form Queue"<<endl;

cout<<"5: List queue"<<endl;

cout<<"6: Change department or exit"<<endl;

choice=ReadNumber();

switch(choice)

{

case 1:

p=InputPatient();

if(p.ID[0])

{

success=q->AddPatientAtEnd(p);

if(success)

{

cout<<"Patient added:"<<endl;

}

else

{

cout<<"Error:The Queue is full.Cannot add poatient"<<endl;

}

OutputPatient(&p);

cout<<"press any key"<<endl;

getch();

}

break;

case 2:

p=InputPatient();

if(p.ID[0])

{

success=q->AddPatientAtBeginning(p);

if(success)

{

cout<<"Patient added:"<<endl;

}

else

{

cout<<"Error:The Queue is full.Cannot add poatient"<<endl;

}

OutputPatient(&p);

cout<<"press any key"<<endl;

getch();

}

break;

case 3:

p=q->GetNextPatient();

if(p.ID[0])

{

cout<<"Patient to operate:"<<endl;

OutputPatient(&p);

}

else

{

cout<<"There is no patient to operate"<<endl;

}

cout<<"press any key"<<endl;

getch();

break;

case 4:

p=InputPatient();

if(p.ID[0])

{

success=q->RemoveDeadPatient(&p);

if(success)

{

cout<<"patient removed:"<<endl;

}

else

{

cout<<"Error: Cannot find patient:"<<endl;

}

OutputPatient(&p);

cout<<"Press any key"<<endl;

getch();

break;

case 5:

q->OutputList();

cout<<" "<<endl;

cout<<"Press any key"<<endl;

getch();

break;

}

}

}

}

int main()

{

int i,MenuChoice=0;

queue departments[3];

strcpy(departments[0].DepartmentName,"Heart Clinic");

strcpy(departments[1].DepartmentName,"Lungs clinic");

strcpy(departments[2].DepartmentName,"Plastic Surgery");

while(MenuChoice!=4)

{

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"\t"" WELCOME TO CITY HOSPITAL:"<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"Please enter your choice:"<<endl;

for(i=0;i<3;i++)

{

cout<<" "<<(i+1)<<": "<<departments[i].DepartmentName<<endl;

}

cout<<" 4: Exit"<<endl;

MenuChoice=ReadNumber();

if(MenuChoice>=1 && MenuChoice<=3)

{

DepartmentMenu(departments+(MenuChoice-1));

}

}

}