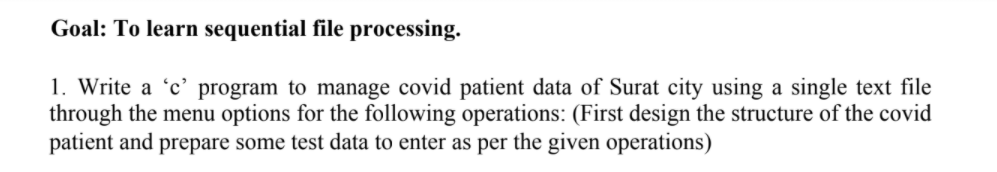
**DATABASE MANAGEMENT SYSTEM**

**ALOK PRASAD**

****

**CODE:**

**Main.cpp**

#include <stdio.h>

#include <string.h>

#include <strings.h>

#include "patient.h"

#include "functions1.h"

#define MAX 256

//functions 4-7

void Ascending(int s);

void descending(int h);

void gender();

void sortFemale();

void sort\_1();

void sort\_2();

void sort\_3();

void sort\_4();

void sort\_5();

void sort\_6();

void sort\_7();

void catergory\_rep(int l);

void func\_A();

void func\_B(int f);

void func\_C(int d);

void count\_Rec();

//Display functions

int menu();

void readFile(char\* file);

int main()

{

char file\_name[20] = "Record.txt";

char file\_name1[20] = "record\_1.txt";

char file\_name2[20] = "record\_2.txt";

char file\_name3[20] = "record\_3.txt";

readFile(file\_name);

while (1) {

int t = menu();

switch (t) {

case 1: {

int a = addFile();

if (a == 2) {

printf("\n\t\t\t Record added!!\n");

}

else {

printf("\n\t\t\t Error! Please repeat");

}

readFile(file\_name);

break;

}

case 2: {

char p[10];

printf("\nEnter patient ID to be del. : ");

scanf("%s", p);

delRec(p);

readFile(file\_name);

break;

}

case 3: {

char p[10];

printf("\nEnter patient ID to be updated : ");

scanf("%s", p);

update(p);

readFile(file\_name);

break;

}

case 4: {

count\_Rec();

break;

}

case 5: {

int l;

printf("enter your choice :");

scanf("%d", &l);

Ascending(l);

readFile(file\_name);

break;

}

case 6: {

int l;

printf("enter your choice :");

scanf("%d", &l);

descending(l);

readFile(file\_name);

break;

}

case 7: {

int l;

printf("enter your choice :");

scanf("%d", &l);

catergory\_rep(l);

readFile(file\_name);

break;

}

case 8: {

gender();

readFile(file\_name1);

printf("\n");

readFile(file\_name2);

printf("\n");

readFile(file\_name3);

printf("\n");

break;

}

case 9: {

return 0;

break;

}

default:

printf("\t\t\tPLEASE SELECT PROPER CHOICE!!!");

}

}

return 0;

}

void readFile(char\* file)

{

char g[20];

struct patients p1;

FILE\* fptr;

fptr = fopen(file, "r+");

printf("\n");

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

short int x = p1.gender;

if (x == 0) {

strcpy(g, "transgender");

}

else if (x == 1) {

strcpy(g, "male");

}

else if (x == 2) {

strcpy(g, "female");

}

else {

strcpy(g, "--------");

}

printf("|\t%s\t|\t%s\t|\t%s\t|\t%d\t|\t%d %d %d\t|\t%d %d %d\t|\t%s\t|\t%d\t|\n", p1.patientCode, p1.firstName, p1.lastName, p1.age, p1.admission.day, p1.admission.month, p1.admission.year, p1.discharge.day, p1.discharge.month, p1.discharge.year, g, p1.area);

printf("\n");

}

fclose(fptr);

}

int menu()

{

int x;

printf("\n\t\t\tMenu\n");

printf("1.ADD A RECORD\n");

printf("2.DELETE A RECORD\n");

printf("3.MODIFY A RECORD\n");

printf("\n4.Reports:\n\tA. DISPLAY COUNT OF PATIENTS \n\tB. DISPLAY COUNT OF PATIENTS BASED ON THE GENDER \n\tC. DISPLAY AGE WISE COUNT OF PATIENTS \n\tD. DIPSPLAY THE COUNT OF PATIENTS AREA WISE\n");

printf("\n5.LIST ALL RECORDS IN AscendingING ORDER\n");

printf("\n6.LIST ALL RECORDS IN DESCENDING ORDER\n");

printf("7.LIST ALL RECORD OF FILE FOR SPECIFIC RANGE: \n\tA.Patient first name starts from your first name letter to next 10 letters \n\tB. Admission date(till next 10 days) \n\tC. Admission month from user given month to next 5 months \n");

printf("8.Male and Female records.\n");

printf("9.EXIT\n");

printf("\nENTER YOUR CHOICE :");

scanf("%d", &x);

return x;

}

void count\_Rec()

{

int arr[MAX];

struct patients p1;

struct patients p2;

char ch;

int count = 0;

int countGM = 0;

int countGF = 0;

int countGT = 0;

int countA = 1;

FILE\* fptr;

fptr = fopen("Record.txt", "r");

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (p1.patientCode) {

count++;

}

}

rewind(fptr);

printf("\nTotal number of patients are :%d \n", count);

if (count > 0) {

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (p1.gender == 0) {

countGT++;

}

if (p1.gender == 1) {

countGM++;

}

if (p1.gender == 2) {

countGF++;

}

}

rewind(fptr);

printf("\n\tMALE : (%d)\n\tFEMALE : (%d)\n\tOTHERS :(%d)\n", countGM, countGF, countGT);

int i = 0;

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

arr[i] = p1.age;

i++;

}

rewind(fptr);

for (i = 0; i < count; i++) {

if (arr[i]) {

for (int j = i + 1; j < count; j++) {

if (arr[j]) {

if (arr[i] == arr[j]) {

countA++;

arr[j] = 0;

}

}

}

printf("\nNumber of patients of age :%d is :(%d)\n", arr[i], countA);

countA = 1;

}

}

rewind(fptr);

i = 0;

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

arr[i] = p1.area;

i++;

}

rewind(fptr);

for (i = 0; i < count; i++) {

if (arr[i]) {

for (int j = i + 1; j < count; j++) {

if (arr[j]) {

if (arr[i] == arr[j]) {

countA++;

arr[j] = 0;

}

}

}

printf("\nNumber of patients of same area:%d is :(%d)\n", arr[i], countA);

countA = 1;

}

}

}

else {

printf("\nNo record found!!!!\n");

}

fclose(fptr);

}

void catergory\_rep(int l)

{

switch (l) {

case 1: {

func\_A();

break;

}

case 2: {

int k;

printf("Enter the date :");

scanf("%d", &k);

func\_B(k);

break;

}

case 3: {

int k;

printf("Enter the month :");

scanf("%d", &k);

func\_C(k);

break;

}

default:

printf("Invalid selection\n");

}

}

void Ascending(int s)

{

switch (s) {

case 1: {

sortFemale();

break;

}

case 2: {

sort\_1();

break;

}

case 3: {

sort\_2();

break;

}

case 4: {

sort\_3();

break;

}

default:

printf("\n Invalid selection\n");

}

}

void descending(int h)

{

switch (h) {

case 1: {

sort\_4();

break;

}

case 2: {

sort\_5();

break;

}

case 3: {

sort\_6();

break;

}

case 4: {

sort\_7();

break;

}

default:

printf("ENTER PROPER CHOICE!!!!!\n");

}

}

/\* Ascendinging start \*/

void sortFemale()

{

struct patients p1;

char fname[MAX][20];

char fname1[20];

FILE\* fptr;

fptr = fopen("Record.txt", "r");

FILE\* fptr1;

fptr1 = fopen("re.txt", "w");

int i = 0;

int count = 0;

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

strcpy(fname[i], p1.firstName);

i++;

count++;

}

rewind(fptr);

for (i = 0; i < count; i++) {

for (int j = i + 1; j < count; j++) {

if (strcmp(fname[i], fname[j]) > 0) {

strcpy(fname1, fname[j]);

strcpy(fname[j], fname[i]);

strcpy(fname[i], fname1);

}

}

}

i = 0;

while (count--) {

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (strcmp(fname[i], p1.firstName) == 0) {

fwrite(&p1, sizeof(struct patients), 1, fptr1);

i++;

}

}

rewind(fptr);

}

fclose(fptr);

fclose(fptr1);

remove("Record.txt");

rename("re.txt", "Record.txt");

}

void sort\_1()

{

struct patients p1;

char fname[MAX][20];

char fname1[20];

FILE\* fptr;

fptr = fopen("Record.txt", "r");

FILE\* fptr1;

fptr1 = fopen("re.txt", "w");

int i = 0;

int count = 0;

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

strcpy(fname[i], p1.lastName);

i++;

count++;

}

rewind(fptr);

for (i = 0; i < count; i++) {

for (int j = i + 1; j < count; j++) {

if (strcmp(fname[i], fname[j]) > 0) {

strcpy(fname1, fname[j]);

strcpy(fname[j], fname[i]);

strcpy(fname[i], fname1);

}

}

}

i = 0;

while (count--) {

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (strcmp(fname[i], p1.lastName) == 0) {

fwrite(&p1, sizeof(struct patients), 1, fptr1);

i++;

}

}

rewind(fptr);

}

fclose(fptr);

fclose(fptr1);

remove("Record.txt");

rename("re.txt", "Record.txt");

}

void sort\_2()

{

struct patients p1;

int fage[20];

int fage1;

FILE\* fptr;

fptr = fopen("Record.txt", "r");

FILE\* fptr1;

fptr1 = fopen("re.txt", "w");

int i = 0;

int count = 0;

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

fage[i] = p1.age;

i++;

count++;

}

rewind(fptr);

for (i = 0; i < count; i++) {

for (int j = i + 1; j < count; j++) {

if (fage[i] > fage[j]) {

fage1 = fage[j];

fage[j] = fage[i];

fage[i] = fage1;

}

}

}

i = 0;

while (count--) {

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (p1.age == fage[i]) {

fwrite(&p1, sizeof(struct patients), 1, fptr1);

i++;

}

}

rewind(fptr);

}

fclose(fptr);

fclose(fptr1);

remove("Record.txt");

rename("re.txt", "Record.txt");

}

void sort\_3()

{

struct patients p1;

int fday[MAX], temp;

int fmonth[MAX];

int fyear[MAX];

int fsum[MAX];

int i = 0;

int count = 0;

FILE\* fptr;

fptr = fopen("Record.txt", "r");

FILE\* fptr1;

fptr1 = fopen("re.txt", "w");

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

fday[i] = p1.admission.day;

fmonth[i] = p1.admission.month;

fyear[i] = p1.admission.year;

fsum[i] = fyear[i] \* 10000 + fmonth[i] \* 100 + fday[i];

i++;

count++;

}

rewind(fptr);

for (i = 0; i < count; i++) {

for (int j = i + 1; j < count; j++) {

if (fsum[i] > fsum[j]) {

temp = fsum[i];

fsum[i] = fsum[j];

fsum[j] = temp;

}

}

}

for (i = 0; i < count; i++) {

fyear[i] = fsum[i] / 10000;

}

for (i = 0; i < count; i++) {

fmonth[i] = (fsum[i] - fyear[i] \* 10000) / 100;

}

for (i = 0; i < count; i++) {

fday[i] = fsum[i] - fyear[i] \* 10000 - fmonth[i] \* 100;

}

i = 0;

while (count--) {

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (p1.admission.year == fyear[i] && p1.admission.month == fmonth[i]

&& p1.admission.day == fday[i]) {

fwrite(&p1, sizeof(struct patients), 1, fptr1);

i++;

}

}

rewind(fptr);

}

fclose(fptr);

fclose(fptr1);

remove("Record.txt");

rename("re.txt", "Record.txt");

}

/\* Ascendinging ends\*/

/\* descending start\*/

void sort\_4()

{

struct patients p1;

char fname[MAX][20];

char fname1[20];

FILE\* fptr;

fptr = fopen("Record.txt", "r");

FILE\* fptr1;

fptr1 = fopen("re.txt", "w");

int i = 0;

int count = 0;

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

strcpy(fname[i], p1.firstName);

i++;

count++;

}

rewind(fptr);

for (i = 0; i < count; i++) {

for (int j = i + 1; j < count; j++) {

if (strcmp(fname[i], fname[j]) < 0) {

strcpy(fname1, fname[j]);

strcpy(fname[j], fname[i]);

strcpy(fname[i], fname1);

}

}

}

i = 0;

while (count--) {

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (strcmp(fname[i], p1.firstName) == 0) {

fwrite(&p1, sizeof(struct patients), 1, fptr1);

i++;

}

}

rewind(fptr);

}

fclose(fptr);

fclose(fptr1);

remove("Record.txt");

rename("re.txt", "Record.txt");

}

void sort\_5()

{

struct patients p1;

char fname[MAX][20];

char fname1[20];

FILE\* fptr;

fptr = fopen("Record.txt", "r");

FILE\* fptr1;

fptr1 = fopen("re.txt", "w");

int i = 0;

int count = 0;

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

strcpy(fname[i], p1.lastName);

i++;

count++;

}

rewind(fptr);

for (i = 0; i < count; i++) {

for (int j = i + 1; j < count; j++) {

if (strcmp(fname[i], fname[j]) < 0) {

strcpy(fname1, fname[j]);

strcpy(fname[j], fname[i]);

strcpy(fname[i], fname1);

}

}

}

i = 0;

while (count--) {

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (strcmp(fname[i], p1.lastName) == 0) {

fwrite(&p1, sizeof(struct patients), 1, fptr1);

i++;

}

}

rewind(fptr);

}

fclose(fptr);

fclose(fptr1);

remove("Record.txt");

rename("re.txt", "Record.txt");

}

void sort\_6()

{

struct patients p1;

int fage[20];

int fage1;

FILE\* fptr;

fptr = fopen("Record.txt", "r");

FILE\* fptr1;

fptr1 = fopen("re.txt", "w");

int i = 0;

int count = 0;

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

fage[i] = p1.age;

i++;

count++;

}

rewind(fptr);

for (i = 0; i < count; i++) {

for (int j = i + 1; j < count; j++) {

if (fage[i] < fage[j]) {

fage1 = fage[j];

fage[j] = fage[i];

fage[i] = fage1;

}

}

}

i = 0;

while (count--) {

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (p1.age == fage[i]) {

fwrite(&p1, sizeof(struct patients), 1, fptr1);

i++;

}

}

rewind(fptr);

}

fclose(fptr);

fclose(fptr1);

remove("Record.txt");

rename("re.txt", "Record.txt");

}

void sort\_7()

{

struct patients p1;

int fday[MAX], temp;

int fmonth[MAX];

int fyear[MAX];

int fsum[MAX];

int i = 0;

int count = 0;

FILE\* fptr;

fptr = fopen("Record.txt", "r");

FILE\* fptr1;

fptr1 = fopen("re.txt", "w");

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

fday[i] = p1.discharge.day;

fmonth[i] = p1.discharge.month;

fyear[i] = p1.discharge.year;

fsum[i] = fyear[i] \* 10000 + fmonth[i] \* 100 + fday[i];

i++;

count++;

}

rewind(fptr);

for (i = 0; i < count; i++) {

for (int j = i + 1; j < count; j++) {

if (fsum[i] < fsum[j]) {

temp = fsum[i];

fsum[i] = fsum[j];

fsum[j] = temp;

}

}

}

for (i = 0; i < count; i++) {

fyear[i] = fsum[i] / 10000;

}

for (i = 0; i < count; i++) {

fmonth[i] = (fsum[i] - fyear[i] \* 10000) / 100;

}

for (i = 0; i < count; i++) {

fday[i] = fsum[i] - fyear[i] \* 10000 - fmonth[i] \* 100;

}

i = 0;

while (count--) {

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (p1.discharge.year == fyear[i] && p1.discharge.month == fmonth[i]

&& p1.discharge.day == fday[i]) {

fwrite(&p1, sizeof(struct patients), 1, fptr1);

i++;

}

}

rewind(fptr);

}

fclose(fptr);

fclose(fptr1);

remove("Record.txt");

rename("re.txt", "Record.txt");

}

/\* descending end\*/

void func\_A()

{

struct patients p1;

char g[20];

FILE\* fptr;

fptr = fopen("Record.txt", "r");

char\* j;

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

j = p1.firstName;

if ((\*j >= 109 && \*j <= 118) || (\*j >= 77 && \*j <= 86)) {

short int x = p1.gender;

if (x == 0) {

strcpy(g, "transgender");

}

else if (x == 1) {

strcpy(g, "male");

}

else if (x == 2) {

strcpy(g, "female");

}

else {

strcpy(g, "--------");

}

printf("\t%s\t|\t%s\t|\t%s\t|\t%d\t|\t%d %d %d\t|\t%d %d %d\t|\t%s\t|\t%d\t|\n", p1.patientCode, p1.firstName, p1.lastName, p1.age, p1.admission.day, p1.admission.month, p1.admission.year, p1.discharge.day, p1.discharge.month, p1.discharge.year, g, p1.area);

printf("\n");

}

}

fclose(fptr);

}

void func\_B(int f)

{

struct patients p1;

FILE\* fptr;

char g[20];

fptr = fopen("Record.txt", "r");

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (p1.admission.day >= f || p1.admission.day < (f + 10) % 32) {

short int x = p1.gender;

if (x == 0) {

strcpy(g, "transgender");

}

else if (x == 1) {

strcpy(g, "male");

}

else if (x == 2) {

strcpy(g, "female");

}

else {

strcpy(g, "--------");

}

printf("\t%s\t|\t%s\t|\t%s\t|\t%d\t|\t%d %d %d\t|\t%d %d %d\t|\t%s\t|\t%d\t|\n", p1.patientCode, p1.firstName, p1.lastName, p1.age, p1.admission.day, p1.admission.month, p1.admission.year, p1.discharge.day, p1.discharge.month, p1.discharge.year, g, p1.area);

printf("\n");

}

}

fclose(fptr);

}

void func\_C(int d)

{

struct patients p1;

FILE\* fptr;

char g[20];

fptr = fopen("Record.txt", "r");

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (p1.admission.month >= d || p1.admission.month < (d + 5) % 13) {

short int x = p1.gender;

if (x == 0) {

strcpy(g, "transgender");

}

else if (x == 1) {

strcpy(g, "male");

}

else if (x == 2) {

strcpy(g, "female");

}

else {

strcpy(g, "--------");

}

printf("\n\t%s\t|\t%s\t|\t%s\t|\t%d\t|\t%d %d %d\t|\t%d %d %d\t|\t%s\t|\t%d\t|\n", p1.patientCode, p1.firstName, p1.lastName, p1.age, p1.admission.day, p1.admission.month, p1.admission.year, p1.discharge.day, p1.discharge.month, p1.discharge.year, g, p1.area);

printf("\n");

}

}

fclose(fptr);

}

void gender()

{

struct patients p1;

FILE\* fptr;

FILE\* fptr1;

FILE\* fptr2;

FILE\* fptr3;

fptr = fopen("Record.txt", "r");

fptr1 = fopen("record\_1.txt", "w");

fptr2 = fopen("record\_2.txt", "w");

fptr3 = fopen("record\_3.txt", "w");

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (p1.gender == 1) {

fwrite(&p1, sizeof(struct patients), 1, fptr1);

}

}

}

**Patient.h**

#include <stdio.h>

struct date {

int day;

int month;

int year;

};

struct patients {

char patientCode[10];

char firstName[20];

char lastName[20];

int age;

int gender;

struct date discharge;

struct date admission;

int area;

};

**Functions1.h**

#include <stdio.h>

int addFile()

{

FILE\* myFile;

myFile = fopen("Record.txt", "a");

if (myFile == NULL) {

printf("file NOT found!\n");

return 1;

}

else {

struct patients p1;

printf("\nEnter patient code: ");

scanf("%s", p1.patientCode);

printf("\nEnter first name: ");

scanf("%s", p1.firstName);

printf("\nEnter lastname: ");

scanf("%s", p1.lastName);

printf("\nEnter AGE: ");

scanf("%d", &p1.age);

printf("\n Enter admission date: ");

scanf("%d %d %d ", &p1.admission.day, &p1.admission.month, &p1.admission.year);

printf("\n Enter the discharge date: ");

scanf(" %d %d %d ", &p1.discharge.day, &p1.discharge.month, &p1.discharge.year);

printf("\n Enter gender (0 for transgender/1 for male / 2 for female): ");

scanf(" % d ", &p1.gender);

printf("\nEnter pin-code: ");

scanf("%d", &p1.area);

fwrite(&p1, sizeof(struct patients), 1, myFile);

fclose(myFile);

return 2;

}

}

void delRec(char\* p)

{

struct patients p1;

FILE \*fptr, \*fptr1, \*fptr2, \*fptr3;

fptr = fopen("Record.txt", "r+");

fptr1 = fopen("p.txt", "w");

if (fptr == NULL || fptr1 == NULL) {

printf("Error");

return;

}

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

if (strcmp(p1.patientCode, p)) {

fwrite(&p1, sizeof(struct patients), 1, fptr1);

}

}

fclose(fptr);

fclose(fptr1);

remove("Record.txt");

rename("p.txt", "Record.txt");

}

void update(char\* a)

{

struct patients p1;

FILE \*fptr, \*fptr1;

fptr = fopen("Record.txt", "r+");

fptr1 = fopen("ascend.txt", "w");

if (fptr == NULL || fptr1 == NULL) {

printf("\nError !!!");

return;

}

while (fread(&p1, sizeof(struct patients), 1, fptr)) {

int x = strcmp(p1.patientCode, a);

if (x == 0) {

printf("\n Enter patient code: ");

scanf("%s", p1.patientCode);

printf("\nEnter the First name of patient: ");

scanf("%s", p1.firstName);

printf("\nEnter the last name of patient: ");

scanf("%s", p1.lastName);

printf("\nEnter age of patient: ");

scanf("%d", &p1.age);

printf("\nEnter the admission date of patient: ");

scanf("%d %d %d ", &p1.admission.day, &p1.admission.month, &p1.admission.year);

printf("\n Enter date of discharge: ");

scanf("%d %d %d ", &p1.discharge.day, &p1.discharge.month, &p1.discharge.year);

printf("\n Enter gender (0: Others, 1:Male, 2:Female) : ");

scanf("%d", p1.gender);

printf("\n Enter pin code: ");

scanf("%d", &p1.area);

}

fwrite(&p1, sizeof(struct patients), 1, fptr1);

}

fclose(fptr);

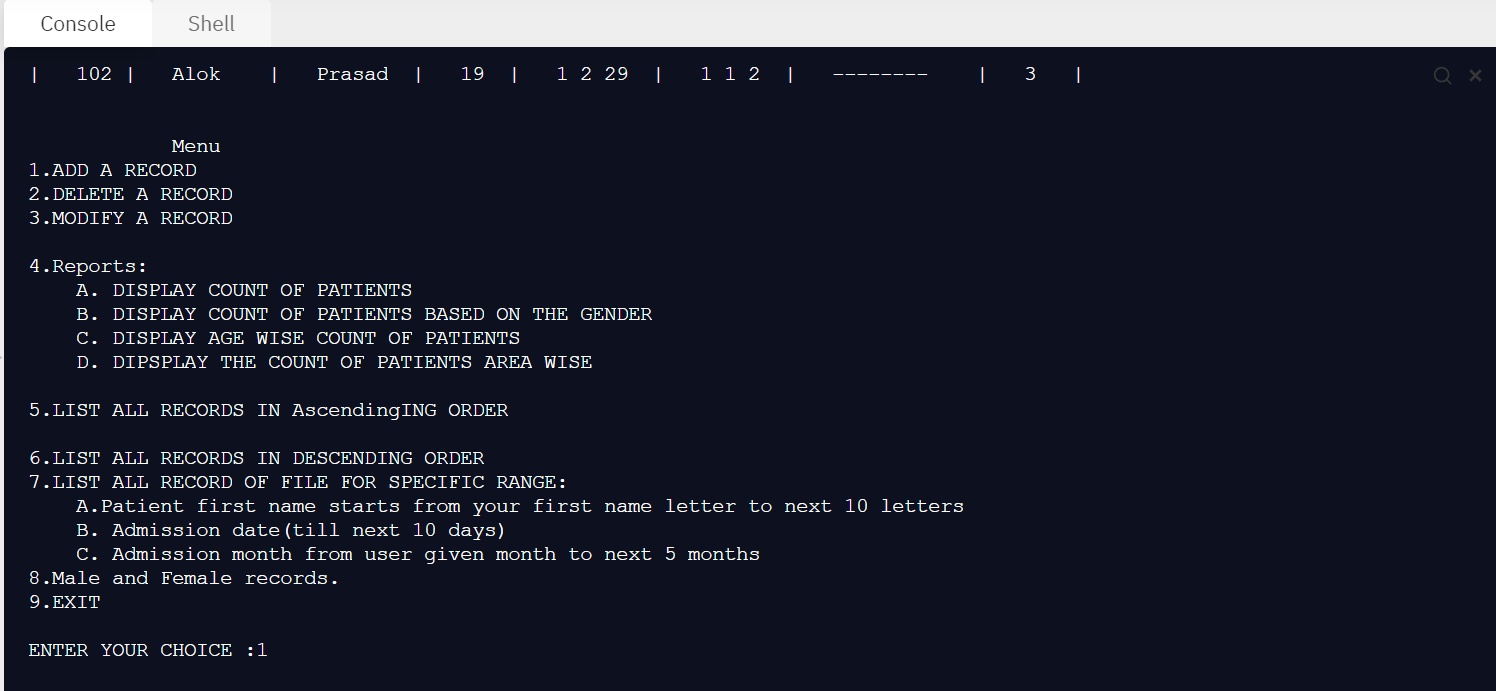
fclose(fptr1);

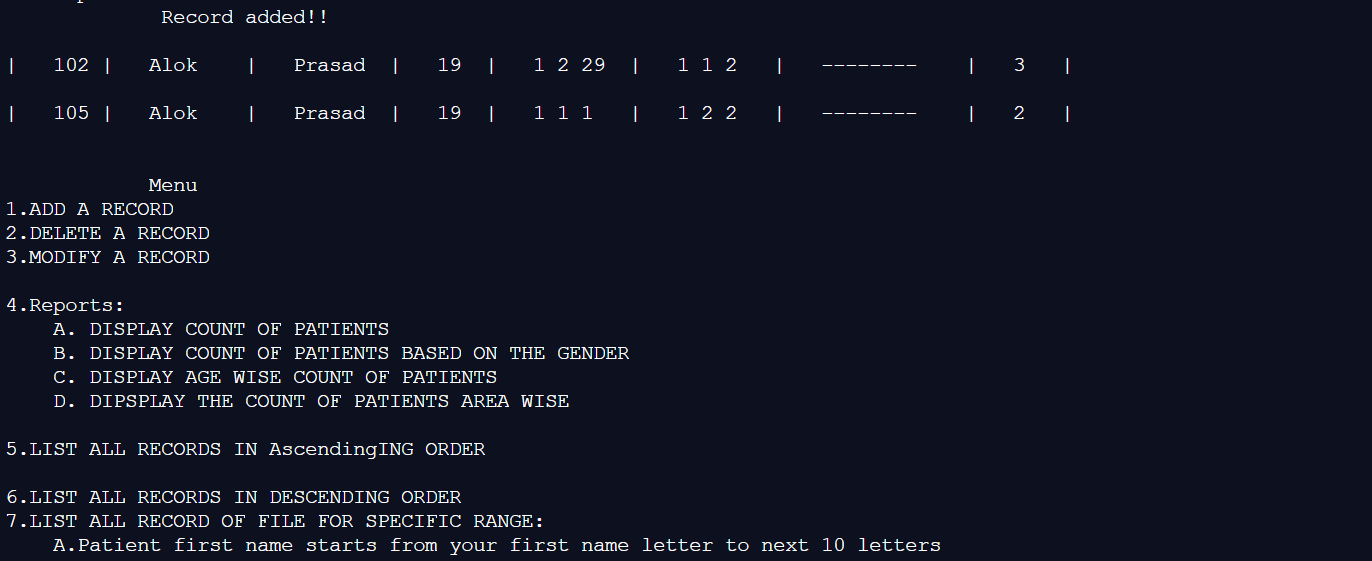
remove("Record.txt");

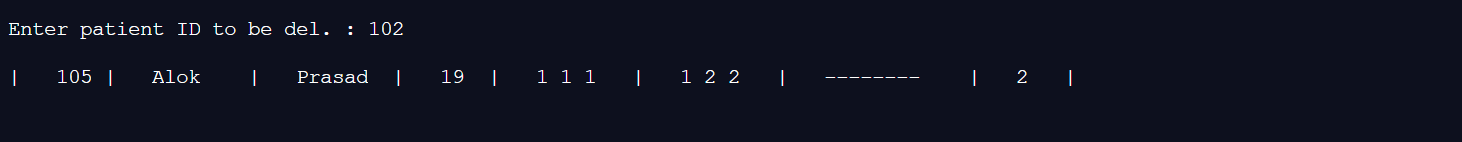
rename("abs.txt", "Record.txt");

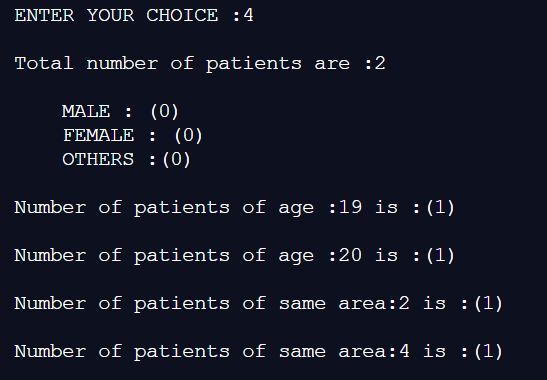
}

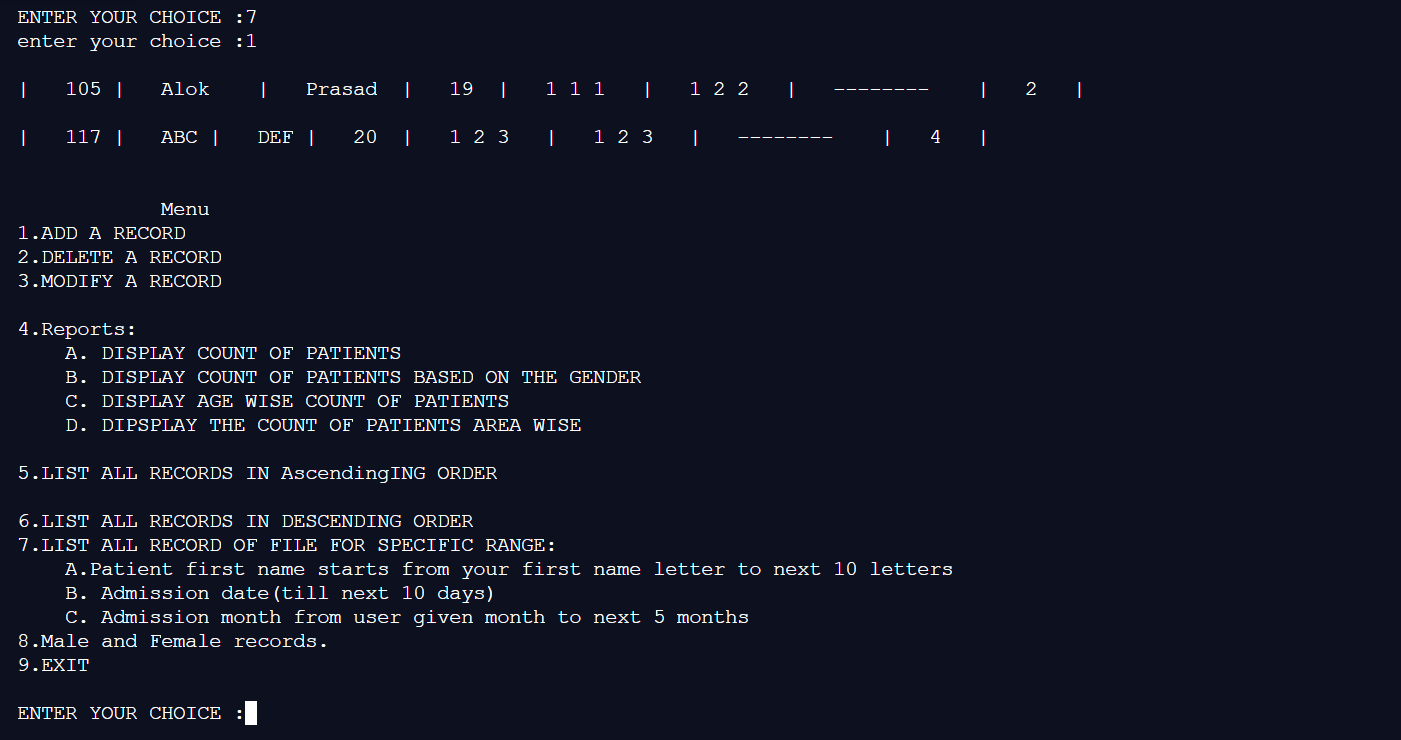
**OUTPUT SCREENSHOTS**

****

****

****

****

****

**-------------------------------------------------------------------------------------------------------------------------------**