***DSA Banana Problem 1***

*Name: Rithvik Rajesh Matta*

*SRN: PES2UG23CS485*

*Class: 3H*

***HOSPITAL MANAGEMENT SYSTEM***

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef enum { head, data } TYPEFIELD;

struct entryNode {

int row;

int col;

int val; // A dummy value to represent the appointment status

char time[10]; // Time of the appointment in HH:MM format

};

typedef struct multiList {

struct multiList \*down;

struct multiList \*right;

TYPEFIELD tag;

union {

struct multiList \*next;

struct entryNode \*entry;

};

} MULTILIST;

MULTILIST\* init(TYPEFIELD tag) {

MULTILIST \*p;

p = (MULTILIST\*)malloc(sizeof(MULTILIST));

p->down = NULL;

p->next = NULL;

p->tag = tag;

if (tag == data) {

p->entry = (struct entryNode\*)malloc(sizeof(struct entryNode));

p->entry->col = 0;

p->entry->row = 0;

p->entry->val = 0;

strcpy(p->entry->time, "");

} else if (tag == head) {

p->next = NULL;

}

p->down = NULL;

p->right = NULL;

return p;

}

int addPatient(MULTILIST \*p) {

MULTILIST \*nn, \*temp = p->down;

nn = init(head);

if (p->down == NULL) {

p->down = nn;

} else {

while (temp->next != NULL)

temp = temp->next;

temp->next = nn;

}

(p->entry->col)++;

return (p->entry->col);

}

int addDoctor(MULTILIST \*p) {

MULTILIST \*nn, \*temp = p->right;

nn = init(head);

if (p->right == NULL) {

p->right = nn;

} else {

while (temp->next != NULL)

temp = temp->next;

temp->next = nn;

}

(p->entry->row)++;

return (p->entry->row);

}

int addAppointment(MULTILIST \*p, int docID, int patID, char \*time)

{

int i;

MULTILIST \*t1 = p->right, \*t2 = p->down, \*nn;

if (docID > p->entry->row || patID > p->entry->col) {

printf("\nInvalid input");

return 0;

}

nn = init(data);

nn->entry->row = patID;

nn->entry->col = docID;

nn->entry->val = 1;

strcpy(nn->entry->time, time);

for (i = 1; i < docID; i++)

t1 = t1->next;

for ( i = 1; i < patID; i++)

t2 = t2->next;

while (t1->down != NULL)

t1 = t1->down;

t1->down = nn;

while (t2->right != NULL)

t2 = t2->right;

t2->right = nn;

printf("\nAppointment added for doctor %d and patient %d at %s", docID, patID, time);

return 1;

}

void deleteAppointment(MULTILIST \*p, int docID, int patID) {

int i;

if (docID > p->entry->row || patID > p->entry->col) {

printf("\nInvalid input");

return;

}

MULTILIST \*t1 = p->right, \*prev1 = NULL;

MULTILIST \*t2 = p->down, \*prev2 = NULL;

for ( i = 1; i < docID; i++) {

prev1 = t1;

t1 = t1->next;

}

for ( i = 1; i < patID; i++) {

prev2 = t2;

t2 = t2->next;

}

MULTILIST \*ap1 = t1->down, \*ap2 = t2->right;

MULTILIST \*apPrev1 = NULL, \*apPrev2 = NULL;

while (ap1 != NULL && (ap1->entry->row != patID || ap1->entry->col != docID)) {

apPrev1 = ap1;

ap1 = ap1->down;

}

while (ap2 != NULL && (ap2->entry->row != patID || ap2->entry->col != docID)) {

apPrev2 = ap2;

ap2 = ap2->right;

}

if (ap1 == ap2 && ap1 != NULL) {

if (apPrev1 != NULL) apPrev1->down = ap1->down;

else t1->down = ap1->down;

if (apPrev2 != NULL) apPrev2->right = ap2->right;

else t2->right = ap2->right;

free(ap1->entry);

free(ap1);

printf("\nAppointment for doctor %d and patient %d deleted.", docID, patID);

} else {

printf("\nAppointment not found.");

}

}

void listAppDoc(MULTILIST \*p, int docID) {

int i;

if (docID > p->entry->row) {

printf("\nInvalid doctor ID");

return;

}

MULTILIST \*t1 = p->right;

for ( i = 1; i < docID; i++)

t1 = t1->next;

printf("\nAppointments for doctor %d:\n", docID);

t1 = t1->down;

while (t1 != NULL) {

printf("Patient %d at %s\n", t1->entry->row, t1->entry->time);

t1 = t1->down;

}

}

void listAppPat(MULTILIST \*p, int patID) {

int i;

if (patID > p->entry->col) {

printf("\nInvalid patient ID");

return;

}

MULTILIST \*t2 = p->down;

for ( i = 1; i < patID; i++)

t2 = t2->next;

printf("\nAppointments for patient %d:\n", patID);

t2 = t2->right;

while (t2 != NULL) {

printf("Doctor %d at %s\n", t2->entry->col, t2->entry->time);

t2 = t2->right;

}

}

int main() {

int ch, docID, patID, cont;

char time[10];

MULTILIST \*p;

p = init(data);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" HOSPITAL MANAGEMENT SYSTEM \n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

do {

printf("\nEnter your choice:");

printf("\n1. Add doctor");

printf("\n2. Add patient");

printf("\n3. Add Appointment");

printf("\n4. Delete Appointment");

printf("\n5. List Appointments for a Doctor");

printf("\n6. List Appointments for a Patient");

printf("\n7. Exit");

printf("\nChoice: ");

scanf("%d", &ch);

switch (ch) {

case 1:

docID = addDoctor(p);

printf("\nDoctor inserted with doctor ID: %d", docID);

break;

case 2:

patID = addPatient(p);

printf("\nPatient inserted with patient ID: %d", patID);

break;

case 3:

printf("\nEnter the doctor ID: ");

scanf("%d", &docID);

printf("\nEnter the patient ID: ");

scanf("%d", &patID);

printf("\nEnter the appointment time (HH:MM): ");

scanf("%s", time);

addAppointment(p, docID, patID, time);

break;

case 4:

printf("\nEnter the doctor ID: ");

scanf("%d", &docID);

printf("\nEnter the patient ID: ");

scanf("%d", &patID);

deleteAppointment(p, docID, patID);

break;

case 5:

printf("\nEnter the doctor ID: ");

scanf("%d", &docID);

listAppDoc(p, docID);

break;

case 6:

printf("\nEnter the patient ID: ");

scanf("%d", &patID);

listAppPat(p, patID);

break;

case 7:

exit(0);

default:

printf("\nInvalid choice");

}

printf("\nDo you want to continue? Press 1 for Yes, 0 for No: ");

scanf("%d", &cont);

} while (cont);

return 0;

}

***OUTPUT:***









