Problem 10:Contest trip

201133216

정유석

• Tomorrow morning Tina must travel from Seoul to Daejeon to compete in the regional programming contest.

• Since she is afraid of arriving late and being excluded from the contest, she is looking for the train which gets her to Daejeon as early as possible.

• However, she dislikes getting to the station too early, so if there are several schedules with the same arrival time then she will choose the one with the latest departure time.

• Tina asks you to help her with her problem. You are given a set of railroad schedules from which you must compute the train with the earliest arrival

time and the fastest connection from one location to another.

• Fortunately, is very experienced in changing trains and can do this instantaneously, i.e., in zero time!

**CODE**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAX\_CITY 100

#define MAX\_TRAIN 100

typedef struct{

char name[50];

}CITY; //city name

typedef struct{

int time[MAX\_CITY];

char Train\_city[MAX\_CITY][MAX\_CITY];

int cnt;

}train; //train information

typedef struct{

int start\_time;

int last\_Time;

}time;

char Departure[50];

char Arrival[50];

int start\_Time;

time total[MAX\_CITY];

CITY city[MAX\_CITY];

train Train[MAX\_TRAIN];

void main(){

int city\_Num,i,j,k,l,m,n,o,p;

int num\_Train,info\_Train;

int check=0;

int min\_Last\_Time,min\_C,min\_Start\_Time;

//Exception

while(1){

printf("Input the number of city : ");

scanf("%d",&city\_Num);

fflush(stdin);

if(city\_Num>1 && city\_Num<101) //1< city <= 100

break;

printf("ERROR ! Reinput ~~ \n");

}

printf("Input the name of cities : \n");//Receive the cities

for(i=0; i<city\_Num; i++)

gets(city[i].name);

while(1){//Receive number of train

printf("Input the number of train : ");

scanf("%d",&num\_Train);

if(num\_Train > 0 && num\_Train < MAX\_TRAIN)

break;

printf("ERROR ! Reinput ~~ \n");

}

//train의 정보 입력받기

for(i=0; i<num\_Train; i++){

while(1){//Exception

printf("Input the number of intersection for train : "); //intersection of train

scanf("%d",&info\_Train);

fflush(stdin);

if(info\_Train < MAX\_CITY && info\_Train > 0){

Train[i].cnt = info\_Train;

break;

}

}

printf("Input the train information : \n");

for(j=0; j<info\_Train; j++){ //Information of each train

int hour;

while(1){ //Exception for train time

scanf("%d %s",&hour,Train[i].Train\_city[j]);

fflush(stdin);

if( (hour/100<24) && (hour/100>=0) && (hour%100<60) && (hour%100>=0) ){

Train[i].time[j] = hour;

break;

}

printf("ERROR ! Reinput ~~ \n");

}

}

}

while(1){ //Exception for Start time

printf("Input the START time : ");

scanf("%d",&start\_Time);

fflush(stdin);

if( (start\_Time/100<24) && (start\_Time/100>=0) && (start\_Time%100<60) && (start\_Time%100>=0) )

break;

printf("ERROR ! Reinput ~~ \n");

}

while(1){//Exception

check = 0;

printf("Input the Departure : \n");

gets(Departure);

printf("Input the Arrival : \n");

gets(Arrival);

if(strcmp(Departure,Arrival) != 0){//Exception - Avoid same with Departure, Arrival name

for(i=0; i<city\_Num; i++){ //Excieption - Check the Departure,Arrival name are correct.

if(strcmp(Departure,city[i].name)==0)

check++;

else if( strcmp(Arrival,city[i].name) ==0 )

check++;

}

}

if(check ==2)//Correct

break;

printf("ERROR ! Reinput ~~ \n");//If occur the error

}

check = 0; //count of total direct

//Dijkstra Algorithm

for(j=0; j<num\_Train; j++){

if(strcmp(Train[j].Train\_city[0],Departure)==0){

if(Train[j].time[0] >= start\_Time){ //FInd same name of Departure

if(strcmp(Train[j].Train\_city[1],Arrival)==0){ //Direct (Departure -> Arrival)

total[check].start\_time = Train[j].time[0];

total[check].last\_Time = Train[j].time[1];

check++;

}

else{ //No Direct

for(k=1; k<Train[j].cnt; k++){

for(n=0; n<num\_Train; n++){

int temp=0; //temp is used to exit the infinite loop

while(1){

temp++;

for(o=0; o<num\_Train; o++){

if(j==n && k==o) //Exception of Same position

continue;

if(strcmp(Train[j].Train\_city[k],Train[n].Train\_city[o])==0 && Train[j].time[k] < Train[n].time[o]){

if(strcmp(Train[n].Train\_city[o+1],Arrival)==0){ //Case by connected train is Direct

total[check].start\_time = Train[j].time[0];

total[check].last\_Time = Train[n].time[o+1];

check++;

temp++;

}

else if(strcmp(Train[n].Train\_city[Train[n].cnt-1],Arrival)==0){ //No Direct

total[check].start\_time = Train[j].time[0];

total[check].last\_Time = Train[n].time[o+1];

check++;

temp++;

}

}

}

if(temp == num\_Train-1)//Exit the infinite loop

break;

}

}

}

}

}

}

else

continue;

}

//Initialize value for finding minimum

min\_Last\_Time = total[0].last\_Time;

min\_Start\_Time = total[0].start\_time;

min\_C = 0;

printf("\n\n");

//Find the minmum arrival cost

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

if(min\_Last\_Time >= total[i].last\_Time){ //Find the minimum arrival time

min\_Last\_Time = total[i].last\_Time;

min\_Start\_Time = total[i].start\_time;

min\_C = i;

if(min\_Start\_Time < total[i].start\_time){ //Find the maximum dparture time

min\_Last\_Time = total[i].last\_Time;

min\_Start\_Time = total[i].start\_time;

min\_C = i;

}

}

}

//Display

printf("Departure %d %s \n",total[min\_C].start\_time,Departure);

printf("Arrival %d %s \n",total[min\_C].last\_Time,Arrival);

}

**Print**





