```cpp
/**
    Grup 4
            Joan Marc Fuentes Soler
            Kirian Rodriguez Alonso
            Oriol Garcia Moreiras
**/
#include <iostream>
#include <cstdlib>
#include <stdlib.h>
#include <stdio.h>
#include <fstream>
#include <map>
#include <queue>
#include <stack>
#include <vector>
#include <list>
#include <cmath>
#include <time.h>
#include <locale.h>

using namespace std;

//Classe Hora
class Hora{

    int h, m, s; //Hora, minuts i segons

public:

    //Constructor per defecte amb l'hora actual, que d√≥na el sistema, llibreria time.h
    Hora(){
        time_t t; //time_t √©s un tipus definit en la llibreria time.h, defineix una tupla de temps
        time( &t );
        struct tm* info = localtime( &t );
        h = info -> tm_hour;
        m = info -> tm_min;
        s = info -> tm_sec;
    }

    //Constructor amb tres par√†metres enters
    Hora(int h, int m, int s){
        this->h=h;
        this->m=m;
        this->s=s;
    }

    //Constructor amb un par√†metre enter quantitat de segons, ha de ser <= 24x3600=84600
    //Se suposa que segons <= 24x3600=84600 segons d'un dia
    Hora(int secs){
        h=secs/3600;
        m=(secs%3600)/60;
        s=(secs%3600)%60;
    }
    //Constructor amb un par√†metre string en format hh:mm:ss o hh-mm-ss
    Hora(string sh){
        string c1 = sh.substr (0,2);//el string dels car√†cters hh
        h=atoi(c1.c_str()); //atoi √©s del C, stoi √©s del C++, la funci√≥ que converteix el string en enter
        string c2 = sh.substr (3,2);//el string dels car√†cters mm
        m=atoi(c2.c_str());
        string c3 = sh.substr (6); //el substring de la 6¬† posici√≥ fins al final
        s=atoi(c3.c_str());
    }

    //Constructor de c√≤pia
    Hora(const Hora& hora)
    {
        h = hora.h;
        m = hora.m;
        s= hora.s;
    }

    //M√®todes consultors
    int getHora(){
        return h;
    }
    int getMinuts(){
        return m;
    }
    int getSegons(){
        return s;
    }

```

```
83        //M√®todes modificadors. Se suposa que h, m i s s√≥n v√†lids
84        void setHora (int hh){
85            h=hh;
86        }
87        void setMinuts (int mm){
88            m=mm;
89        }
90        void setSegons (int ss){
91            s=ss;
92        }
93
94        //Incrementar la data amb una quantitat de segons
95        void incrementaHora (int ss){
96            int segs = s + ss; //sumem els segons totals resultants
97            if (segs > 59) {
98                int hh=segs/3600;
99                h+=hh;
100               int mm =(segs - hh*3600)/60;
101               m+=mm;
102               if (m > 59)
103               {
104                   h++;
105                   m-=60;
106               }
107               s=segs- hh*3600-mm*60;
108           }
109       }
110
111      // L'operador d'assignaci√≥
112       Hora& operator = (const Hora& hora)
113       {
114           if (this != &hora)
115           {
116               h = hora.h;
117               m = hora.m;
118               s= hora.s;
119           }
120           return (*this);
121       }
122
123       //L'operador incrementa l'hora amb una quantitat de segons ss
124       friend Hora& operator + (Hora& H, int ss)
125       {
126           int segs= H.s + ss;
127           if (segs > 59) {
128               int hh=segs/3600;
129               H.h+=hh;
130               int mm =(segs - hh*3600)/60;
131               H.m+=mm;
132               if (H.m > 59)
133               {
134                   H.h++;
135                   H.m-=60;
136               }
137               H.s=segs- hh*3600-mm*60;
138           }
139           return H;
140       }
141
142       //L'operador resta l'hora que s'esta construint amb una hora donada H. Se suposa que les
      dues hores estan dintre del mateix dia
143       friend Hora operator - (Hora H1, Hora H2)
144       {
145           int s1 = H1.getHora()*3600 + H1.getMinuts()*60 + H1.getSegons();
146            int s2 = H2.getHora()*3600 + H2.getMinuts()*60 + H2.getSegons();
147
148           int s = abs(s1-s2);
149
150           return Hora(s);
151       }
152
153       //Operador < per comparar dues hores
154       friend bool operator < (Hora& h1, Hora& h2)
155       {
156           if ( h1.h < h2.h)
157           {
158               return true;
159           }
160           else if (h1.h == h2.h and h1.m < h2.m)
161           {
162               return true;
163           }
164           else if ( h1.h == h2.h and h1.m == h2.m and h1.s < h2.s)
165           {
```

```
166                 return true;
167             }
168             return false;
169         }
170 
171         //Comparar dues hores ==
172         friend bool operator == (Hora& h1, Hora& h2)
173         {
174             return ( h1.h == h2.h and h1.m == h2.m and h1.s == h2.s);
175         }
176 
177         //Llegir una hora per canal d'entrada
178         friend istream& operator >> (istream& is, Hora& H)
179         {
180             is >> H.h;
181             is >> H.m;
182             is >> H.s;
183 
184             return is;
185         }
186 
187         //Escriure una data per canal de sortida hh:mm:ss format 24h
188         friend ostream& operator << (ostream& os, Hora hora)
189         {
190             os << "Hora: ";
191             if (hora.h < 10){
192                 os << "0";
193             }
194             os << hora.h <<":";
195 
196             if (hora.m < 10){
197                 os << "0";
198             }
199             os<< hora.m <<":";
200 
201             if (hora.s < 10){
202                 os << "0";
203             }
204             os<< hora.s << endl;
205 
206             return os;
207         }
208 
209     };
210 
211     //Classe Data
212     class Data{
213 
214         int dia, mes, any;
215 
216     public:
217 
218         //Constructor per defecte amb la data actual, que d√≥na el sistema, llibreria time.h
219         Data(){
220             time_t t; //time.t √©s una tupla de temps (dia, mes any, hora, mins, segs) definit en
    time.h
221             time( &t );
222             struct tm* info = localtime( &t );
223             dia = info -> tm_mday;
224             mes = info -> tm_mon + 1;
225             any = 1900+ info -> tm_year; // l'any es comen√ßa a comptar a partir del 1900
226         }
227 
228         //Constructor amb tres par√†metres enters
229         Data(int d, int m, int a){
230             dia=d;
231             mes=m;
232             any=a;
233         }
234 
235         //Constructor amb un par√†metre enter ddmmaaaa
236         Data(int ddmmaaaa){
237             dia=ddmmaaaa/1000000;
238             mes=(ddmmaaaa/10000)%100;
239             any=ddmmaaaa%10000;
240         }
241 
242         //Constructor amb un par√†metre string en format dd:mm:aaaa o dd-mm-aaaa
243         //stoi √©s una funci√≥ per convertir un string (p.ex "12") en un enter (p.ex. 12)
244         //Pels compiladors de C++ 11 o posterior, sin√≥ utilitzeu atoi
245         Data(string sd){
246             string c1 = sd.substr (0,2);//el string dels car√†cters dd
247             dia=atoi(c1.c_str()); //Per
248             string c2 = sd.substr (3,2);//el string dels car√†cters mm
```

```
249             mes=atoi(c2.c_str());
250             string c3 = sd.substr (6); //el substring de la 6‡† posiciÛ fins al final
251             any=atoi(c3.c_str());
252         }
253
254         //Constructor de c√≤pia
255         Data(const Data& data)
256         {
257             dia = data.dia;
258             mes = data.mes;
259             any = data.any;
260         }
261
262         //M√®todes consultors
263         int getDia(){
264             return dia;
265         }
266         int getMes(){
267             return mes;
268         }
269         int getAny(){
270             return any;
271         }
272
273         //Comprovar si la data que s'esta construint √©s de traspÈs
274         bool de_traspas(int any) const
275         {
276             return ((any % 4 == 0 && any % 100 != 0) or any%400 == 0);
277         }
278
279         //Consultar dies d'un mes i any donats. M√®tode privat. Nom√©s s'utilitzar√† en la classe
280         int dias_mes(int mes, int any) const
281         {
282                 int dies = 31;
283                 if (mes == 4 or mes == 6 or mes == 9 or mes == 11) {
284                     dies = 30;
285                 }
286                 else if (mes == 2) {
287                     if (de_traspas(any)) {
288                         dies = 29;
289                     }
290                     else {
291                         dies = 28;
292                     }
293                 }
294                 return dies;
295             }
296
297         //Comptar els dies des d'1 de gener fins al mes (incl√≤s) de una data donada
298         int compta_dias() const
299         {
300             int sd = 0; //suma de dies
301             for (int i = 1; i<= mes; i++)
302             {
303                 sd = sd + dias_mes(i,any);
304             }
305             return sd;
306         }
307         //Diferencia de dies entre dues dates, la data D i la que s'est√† construint
308         int dif_dias(const Data& D) // se suposa que les dates s√≥n del mateix any
309         {
310             //comptem els dies fins el mes (incl√≤s) i restem els dies que sobren del mes
311             return ((D.compta_dias()-(dias_mes(D.mes,D.any)-D.dia)) -
    (compta_dias()-(dias_mes(mes,any)-dia)));
312         }
313
314         //M√®todes modificadors. Se suposa que d, m i a s√≥n v√†lids
315         void setDia (int d){
316             dia=d;
317         }
318         void setMes (int m){
319             mes=m;
320         }
321         void setAny (int a){
322             any=a;
323         }
324
325         //Incrementar la data amb una quantitat de dies
326         void incrementaData (int dd){
327             dia=dia+dd;
328             if (dia > dias_mes(mes, any)) {
329                 dia = dia-dias_mes(mes,any);
330                 mes++;
331                 if (mes > 12) {
```

```
                mes = 1;
                any++;
            }
        }
    }

    //Operador d'assignaci√≥
    Data& operator = (const Data& data)
    {
        if (this != &data)
        {
            dia = data.dia;
            mes = data.mes;
            any = data.any;
        }
        return (*this);
    }

    //Diferencia de dies entre dues dates, la data D i la que s'est√† construint
    int operator - (const Data& D) // se suposa que les dates s√≥n del mateix any
    {
        //comptem els dies fins el mes (incl√≤s) i restem els dies que sobren del mes
        return ((compta_dias()-(dias_mes(mes,any)-dia))
-(D.compta_dias()-(dias_mes(D.mes,D.any)-D.dia)));
    }

    //Comparar dues dates <
    friend bool operator < (Data& d1, Data& d2)
    {
        return (d1.any*10000+d1.mes*100+d1.dia < d2.any*10000+d2.mes*100+d2.dia);
    }

    //Comparar dues dates ==
    friend bool operator == (Data& d1, Data& d2)
    {
        return (d1.any*10000+d1.mes*100+d1.dia == d2.any*10000+d2.mes*100+d2.dia);
    }

    //Llegir una data per canal d'entrada
    friend istream& operator >> (istream& is, Data& D)
    {
        is >> D.dia;
        is >> D.mes;
        is >> D.any;

        return is;
    }

    //Escriure una data per canal de sortida dd/mm/aaaa
    friend ostream& operator << (ostream& os, Data d)
    {
        os << "Data: ";
        if (d.dia < 10){
            os << "0";
        }
        os << d.dia <<"/";

        if (d.mes < 10){
            os << "0";
        }
        os<< d.mes << "/" << d.any<<endl;

        return os;
    }

};

//La nova classe DataHora
class DataHora{

    Data data;
    Hora hora;

public:

    //Constructor per defecte. Cridar√† els constructors per defecte de Data i Hora, aix√≠ donos crear√†
    //la Data i Hora actuals
    DataHora(){}

    //Constructor amb 6 par√†metres dd mm aaaa hh mins segs
    //Cridar√† els constructors respectius de Data i Hora
    DataHora(int dd, int mm, int aaaa, int hh, int mins, int
segs):data(dd,mm,aaaa),hora(hh,mins,segs){}

```

```
414         //Cosntructor amb dos par√†metres
415         DataHora(Data d,Hora h):data(d),hora(h){}
416
417         //Constructor amb dues cadenes dd-mm-aaaa hh:mins:segs
418         DataHora(string sdata, string shora):data(sdata),hora(shora){}
419
420         //Constructor de c√≤pia
421         DataHora(const DataHora& dh)
422         {
423             data = dh.data;
424             hora = dh.hora;
425         }
426
427         //M√®todes consultors
428         Data& getData()
429         {
430             return data;
431         }
432
433         Hora& getHora()
434         {
435             return hora;
436         }
437
438         bool detraspas()
439         {
440             return data.de_traspas(data.getAny());
441         }
442
443         int getDia()
444         {
445             return data.getDia();
446         }
447
448         int getMes(){
449             return data.getMes();
450         }
451
452         int getAny(){
453             return data.getAny();
454         }
455
456         int getLHora(){
457             return hora.getHora();
458         }
459         int getMinuts(){
460             return hora.getMinuts();
461         }
462         int getSegons(){
463             return hora.getSegons();
464         }
465
466         //Aquest m√®tode retorna true si dues DataHora coincideixen en la mateixa data
467         //Podria ser √∫til si es vol saber si dos esdeveniments diferents han ocorregut el mateix dia
468         friend bool mateixaData(DataHora& dh1, DataHora& dh2)
469         {
470             return dh1.data == dh2.data;
471         }
472
473         //Aquest m√®tode retorna true si dues DataHora coincideixen en la mateixa hora
474         //Podria ser √∫til si es vol saber si dos esdeveniments diferents han ocorregut en la
    mateixa hora
475         friend bool mateixaHora(DataHora& dh1, DataHora& dh2)
476         {
477             return dh1.hora == dh2.hora;
478         }
479         //M√®todes modificadors. Se suposa que d, m i a s√≥n v√†lids
480         void setDia (int d){
481             data.setDia(d);
482         }
483         void setMes (int m){
484             data.setMes(m);
485         }
486         void setAny (int a){
487             data.setAny(a);
488         }
489
490         //M√®todes modificadors. Se suposa que h, m i s s√≥n v√†lids
491         void setHora (int hh){
492             hora.setHora(hh);
493         }
494         void setMinuts (int mm){
495             hora.setMinuts(mm);
496         }
```

```
497       void setSegons (int ss){
498           hora.setSegons(ss);
499       }
500
501       void incrementaData(int dd)
502       {
503           data.incrementaData(dd);
504       }
505
506       void setDataHoraActual()
507       {
508           time_t t; //time_t Ès un tipus definit en la llibreria time.h, defineix una tupla de
      temps
509           time( &t );
510           struct tm* info = localtime( &t );
511           hora.setHora(info -> tm_hour);
512           hora.setMinuts(info -> tm_min);
513           hora.setSegons(info -> tm_sec);
514           data.setDia(info -> tm_mday);
515           data.setMes(info -> tm_mon + 1);
516           data.setAny(1900+ info -> tm_year); // l'any es comenÃ§a a comptar a partir del 1900
517       }
518
519       //Reset data a 01/01/1900
520       void resetDataHora()
521       {
522           data.setDia(1);
523           data.setMes(1);
524           data.setAny(1900);
525           hora.setHora(0);
526           hora.setMinuts(0);
527           hora.setSegons(0);
528       }
529       //Operadors
530
531       //Operador + per sumar una quantitat de segons.
532       //Pot provocar un canvi de la hora i data en cadena
533       friend DataHora& operator += (DataHora& dh, int ss)
534       {
535           int nousSegs = dh.getSegons() + ss; //segons totals dESArÃ©s d'afegir
536
537           if (nousSegs > 59)
538           {
539               int nousMins = nousSegs/60;//minuts addicionals
540               nousSegs = nousSegs - nousMins*60;
541               dh.setSegons(nousSegs); //segons resultants finals
542
543               nousMins = nousMins + dh.getMinuts(); //nous minuts sumant els existents
544               if (nousMins > 59)
545               {
546                   int novesH = nousMins/60;
547                   nousMins = nousMins - novesH*60;
548                   dh.setMinuts(nousMins);//minuts resultants finals
549
550                   novesH = novesH + dh.getLHora();
551                   if (novesH > 24)
552                   {
553                       int nousD = novesH/24;
554                       novesH = novesH - nousD*24;
555                       dh.setHora(novesH); //hores resultants finals
556
557                       dh.incrementaData(nousD);
558                   }
559                   else
560                   {
561                       dh.setHora(novesH);
562                   }
563               }
564               else
565               {
566                   dh.setMinuts(nousMins);
567               }
568           }
569           else
570           {
571               dh.setSegons(nousSegs);
572           }
573
574           return dh;
575       }
576
577       //Operador d'assignaciÃ²
578       DataHora& operator = (const DataHora& dh)
579       {
```

```
580             if (this != &dh)
581             {
582                 data = dh.data;
583                 hora = dh.hora;
584             }
585             return (*this);
586         }
587 
588         //Comparar dues dates <
589         friend bool operator < (DataHora& dh1, DataHora& dh2)
590         {
591             return ((dh1.data < dh2.data) or ((dh1.data == dh2.data) and (dh1.hora < dh2.hora)));
592         }
593 
594         //Comparar dues dates ==
595         friend bool operator == (DataHora& dh1, DataHora& dh2)
596         {
597             return ((dh1.data == dh2.data) and (dh1.hora == dh2.hora));
598         }
599 
600         //Llegir una data per canal d'entrada
601         friend istream& operator >> (istream& is, DataHora& dh)
602         {
603             is >> dh.data;
604             is >> dh.hora;
605 
606             return is;
607         }
608 
609         //Escriure una data per canal de sortida dd/mm/aaaa
610         friend ostream& operator << (ostream& os, DataHora dh)
611         {
612             os << dh.data << dh.hora;
613 
614             return os;
615         }
616 
617     };
618 
619     //Classe Persona
620     class Persona{
621 
622         //-------------->Dades<--------------//
623         protected:
624         string Nom,Cognom,DNI;
625         int Edat;
626         char Sexe;
627     public:
628         //--------------->API<---------------//
629 
630         //----------->Constructors<----------//
631 
632         //Constructor per defecte
633         Persona(){}
634         //Constructor amb parametres
635         Persona(string pNom,string pCognom,string pDNI,int pEdat,char pSexe){
636 
637                 Nom = pNom;
638                 Cognom = pCognom;
639                 DNI = pDNI;
640                 Edat = pEdat;
641                 Sexe = pSexe;
642         }
643         //Constructor amb parametres des d'un fitxer
644         Persona(ifstream& fpersona){
645 
646             fpersona >> Nom;
647             fpersona >> Cognom;
648             fpersona >> DNI;
649             fpersona >> Edat;
650             fpersona >> Sexe;
651         }
652         //Constructor de copia
653         Persona(const Persona& P){
654 
655                 Nom = P.Nom;
656                 Cognom = P.Cognom;
657                 DNI = P.DNI;
658                 Edat = P.Edat;
659                 Sexe = P.Sexe;
660         }
661         //MËtodes consultors getX
662         string getNom()const{
663 
```

```
664             return Nom;
665         }
666         string getCognom() const{
667 
668             return Cognom;
669         }
670         string getDNI()const{
671 
672             return DNI;
673         }
674         int getEdat() const{
675 
676             return Edat;
677         }
678         char getSexe() const{
679 
680             return Sexe;
681         }
682         //MËtodes modificadors setX
683          void setNom(string pNom){
684 
685             pNom = Nom;
686         }
687         void  setCognom(string pCognom) {
688 
689             pCognom = Cognom;
690         }
691         void  setDNI(string pDNI){
692 
693             pDNI = DNI;
694         }
695         void  setEdat(int pEdat) {
696 
697             pEdat = Edat;
698         }
699         void  setSexe(char pSexe) {
700 
701             pSexe = Sexe;
702         }
703         //Operador  d'asignaciÛ
704         Persona& operator = (const Persona& P){
705 
706             if(this !=&P){
707 
708                 Nom = P.Nom;
709                 Cognom = P.Cognom;
710                 DNI = P.DNI;
711                 Edat = P.Edat;
712                 Sexe = P.Sexe;
713 
714             }
715             return (*this);
716         }
717         //Operadors relacionals
718 
719         //Operador de ==
720         friend bool operator == (Persona& P1 , Persona&  P2){
721 
722             return((P1.Nom == P2.Nom) and (P1.Cognom == P2.Cognom) and (P1.DNI == P2.DNI) and
    (P1.Edat == P2.Edat) and (P1.Sexe == P2.Sexe));
723 
724         }
725         friend bool operator < (Persona& P1 , Persona&  P2){
726 
727             return(P1.Edat < P2.Edat);
728         }
729         friend bool operator > (Persona& P1 , Persona&  P2){
730 
731             return(P1.Edat > P2.Edat);
732         }
733         // Operadors d'extracciÛ >>
734 
735         friend istream& operator >> (istream& is, Persona &P){
736 
737             is >> P.Nom;
738             is >> P.Cognom;
739             is >> P.DNI;
740             is >> P.Edat;
741             is >> P.Sexe;
742 
743             return is;
744         }
745         // Operadors d'inserciÛ <<
746 
```

```
747         friend ostream& operator << (ostream& os, Persona& P){
748 
749             os << "Nom: "<< P.Nom << endl;
750             os << "Cognom: "<< P.Cognom << endl;
751             os << "DNI: " << P.DNI << endl;
752             os << "Edat: " << P.Edat << endl;
753             os << "Sexe: " << P.Sexe << endl;
754 
755             return os;
756         }
757 
758     };
759 
760     //Classe Alumnes
761     class Alumne : public Persona{
762 
763         //-------------->Dades<--------------//
764         string Aula,DNIA;
765         int Curs;
766         DataHora HALE; //DataHora del Alumne que entra
767         DataHora HALS; //DataHora del Alumne que entra
768         double Temp;
769     public:
770         //--------------->API<---------------//
771 
772         //----------->Constructors<----------//
773 
774         //Constructor per defecte
775         Alumne(){}
776         //Constructor amb parametres
777         Alumne(Persona P,string pDNIA,string pAula,int pCurs,DataHora pHALE,DataHora pHALS, double
    pTemp):Persona(P){
778 
779             DNIA = pDNIA;
780             Aula = pAula;
781             Curs = pCurs;
782             HALE = pHALE;
783             HALS = pHALS;
784             Temp = pTemp;
785 
786         }
787         //Constructor amb parametres des d'un fitxer
788         Alumne(Persona P,ifstream& falumne):Persona(P){
789 
790             falumne >> DNIA;
791             falumne >> Aula;
792             falumne >> Curs;
793             falumne >> HALE;
794             falumne >> HALS;
795             falumne >> Temp;
796 
797         }
798         //Constructor de copia
799         Alumne(const Alumne& AL):Persona(AL){
800 
801             DNIA = AL.DNIA;
802             Aula = AL.Aula;
803             Curs = AL.Curs;
804             HALE = AL.HALE;
805             HALS = AL.HALS;
806             Temp = AL.Temp;
807 
808         }
809         //MËtodes consultors getX
810         string getDNIA()const{
811 
812             return DNIA;
813         }
814         string getAula()const{
815 
816             return Aula;
817         }
818         int getCurs() const{
819 
820             return Curs;
821         }
822         DataHora getHoraALE()const{
823 
824             return HALE;
825         }
826         DataHora getHoraALS()const{
827 
828             return HALS;
829         }
```

```
830         double getTemp()const{
831 
832             return Temp;
833         }
834         //MËtodes modificadors setX
835         void setDNIA(string pDNIA){
836 
837             pDNIA = DNIA;
838         }
839         void setAula(string pAula){
840 
841             pAula = Aula;
842         }
843         void setCurs(int pCurs){
844 
845             pCurs = Curs;
846         }
847         void setHoraALE(DataHora pHALE){
848 
849             pHALE = HALE;
850         }
851         void setHoraALS(DataHora pHALS){
852 
853             pHALS = HALS;
854         }
855         void setTemp(double pTemp){
856 
857             pTemp = Temp;
858         }
859         //Operador  d'asignaciÛ
860         Alumne& operator = (const Alumne& AL){
861 
862             if(this !=&AL){
863 
864             Persona::operator = (AL);
865             DNIA = AL.DNIA;
866             Aula = AL.Aula;
867             Curs = AL.Curs;
868             HALE = AL.HALE;
869             HALS = AL.HALS;
870             Temp = AL.Temp;
871 
872             }
873             return (*this);
874         }
875         //Operadors relacionals
876         friend bool operator == (Alumne& AL1 , Alumne&  AL2){
877 
878             return((AL1.DNIA == AL2.DNIA) and (AL1.Aula == AL2.Aula)and (AL1.Curs == AL2.Curs) and
    (AL1.HALE == AL2.HALE)and (AL1.HALS == AL2.HALS) and (AL1.Temp = AL2.Temp));
879 
880         }
881         friend bool operator < (Alumne& AL1 , Alumne&  AL2){return true;}
882         friend bool operator > (Alumne& AL1 , Alumne&  AL2){return true;}
883 
884         // Operadors d'extraccciÛ >>
885 
886         friend istream& operator >> (istream& is, Alumne &AL){
887 
888             Persona P; // La part Persona de A
889             //is >> P;
890 
891             string Aula,DNIA;
892             int Curs;
893             DataHora HALE;
894             DataHora HALS;
895             double Temp;
896 
897             is >> DNIA;
898             is >> Aula;
899             is >> Curs;
900             is >> HALE;
901             is >> HALS;
902             is >> Temp;
903 
904             Alumne PAL (P,DNIA,Aula,Curs,HALE,HALS,Temp);
905             AL = PAL;
906 
907             return is;
908         }
909         // Operadors d'insercciÛ <<
910 
911         friend ostream& operator << (ostream& os, Alumne& AL){
912 
```

```
913             Persona P = (Persona) AL; //Extrau la part de la persona de A
914             os << "Fitxa de l'Alumne: " << endl;
915             //os << P;
916             os << "------------------ " << endl;
917             os << "DNI: " << AL.DNIA << endl;
918             os << "Aula: " << AL.Aula << endl;
919             os << "Curs: " << AL.Curs << endl;
920             os << "Entrada: " << endl;
921             os << AL.HALE << endl;
922             os << "Sortida: " << endl;
923             os << AL.HALS;
924             os << "Temperatura: " << AL.Temp << endl;
925             os << "-------------------" << endl;
926 
927             return os;
928         }
929 
930     };
931 
932     //Fem un petit struct per representar les dades d'entrada i sortida d'un alumne en una aula en una hora d'entrada i sortida donades (es suposa al llarg del dia)
933     struct EntSort{
934        DataHora DHE; //DataHora de Entrada
935        DataHora DHS; //DataHora de Sortida
936        string DNI; //Una sola entrada/salida per un sol alumne! Mireu la classe Aula per la mem dinàmica
937 
938         EntSort(){}
939 
940         EntSort(string pDNI, DataHora pDHE, DataHora pDHS){
941             DNI = pDNI;
942             DHE = pDHE;
943             DHS = pDHS;
944         }
945 
946         EntSort(const EntSort& ES){
947             DNI = ES.DNI;
948             DHE = ES.DHE;
949             DHS = ES.DHS;
950         }
951 
952         EntSort& operator = (const EntSort& ES){
953             if (this != &ES){
954                 DHE = ES.DHE;
955                 DHS = ES.DHS;
956                 DNI = ES.DNI;
957             }
958             return (*this);
959         }
960 
961         friend ostream& operator << (ostream& os, const EntSort& ES){
962             cout << "E/S:" << endl;
963             cout << "----------------------" << endl;
964             cout << "HE: " << ES.DHE; //DataHora d'Entrada
965             cout << "HS: " << ES.DHS; //DataHora d'Entrada
966             cout << "DNI: " << ES.DNI << endl;
967             cout << "----------------------" << endl;
968 
969             return os;
970         }
971     };
972 
973     //Classe Aula
974     class Aula {
975     //--------------->Dades<--------------//
976 
977         int CMAX; //Capacitat Maxima d'alumnes en una aula
978         DataHora DHE; //DataHora de Entrada
979         DataHora DHS; //DataHora de Sortida
980         string nomAula;
981 
982         //Mem. dinàmica
983         EntSort* ESs; // Quantitat indeterminada d'entrades i sortides d'alumnes a l'aula
984         int nES; //dimensiÛ de la taula
985 
986 
987          //Resta entre datahora de sortida i datahora d'entrada HAUS - HAUE
988         void dias_horas_minuts(DataHora HAUS, DataHora HAUE, int& DD, int& HH, int& MM)
989         {
990             //Dates diferenta
991             if (!(HAUS.getData() == HAUE.getData()))
992             {
993                 DD = HAUS.getData() - HAUE.getData() - 1;
994 
```

```
995             Hora Mitjanit1(0,0,0);
996             Hora Mitjanit2(24,0,0);
997
998             Hora hores1 = HAUS.getHora() - Mitjanit1;
999             Hora hores2 = Mitjanit2 - HAUE.getHora();
1000
1001            int Hor = hores1.getHora() + hores2.getHora();
1002            convertir(Hor,DD,HH);
1003
1004            MM =  hores1.getMinuts() + hores2.getMinuts();
1005        }
1006        else
1007        {
1008            DD = 0;
1009            Hora hdif = HAUS.getHora() - HAUE.getHora();
1010            HH = hdif.getHora();
1011            MM = hdif.getMinuts();
1012        }
1013    }
1014    //Convertir una quantitat de minuts en dies, hores, minuts
1015    void DiesHoresMins(int minuts, int& Dies, int& Hores, int& Mins)
1016    {
1017        Dies  = minuts / 1440;
1018        Hores = (minuts % 1440) / 60;
1019        Mins  = (minuts % 1440) % 60;
1020    }
1021
1022    void convertir(int Hor, int& DD, int& HH){
1023        DD = Hor / 24;
1024        HH = Hor % 24;
1025    }
1026
1027  public:
1028    //--------------->API<---------------//
1029
1030    //----------->Constructors<----------//
1031
1032    //Constructor per defecte
1033    Aula(){}
1034
1035    //Amb parametres, dades llegides des d'un fitxer
1036    Aula(ifstream& faula){
1037
1038        faula >> CMAX; //llegim capacitat màxima
1039
1040        faula >> nES; //llegim el nombre d'entrades i sortides a l'aula
1041
1042        //Adquirim memòria
1043        ESs = new EntSort[nES];
1044
1045        string nomAula, DNIA;
1046        int dde, mme, aaaae;
1047        int hhe, mine, sece;
1048        int dds, mms, aaaas;
1049        int hhs, mins, secs;
1050        int i = 0; //ens en la taula dinàmica
1051        while (faula >> nomAula >> DNIA >> dde >> mme >> aaaae >> hhe >> mine >> sece >> dds >>
    mms  >> aaaas >> hhs >> mins >> secs)
1052        {
1053            DataHora DHENT(dde, mme, aaaae, hhe, mine, sece);
1054            DataHora DHSORT(dds, mms, aaaas,hhs,mins,secs);
1055
1056            EntSort EnSr(DNIA,DHENT,DHSORT);
1057
1058            ESs[i] = EnSr;
1059            i++;
1060        }
1061    }
1062    //Constructor amb parametres
1063    //HO PODEU ARREGLAR EN BASE DE L'ANTERIOR CONSTR.
1064    Aula(int pCMAX, string pnomAula,ifstream& faula){
1065
1066        CMAX = pCMAX;
1067        nomAula = pnomAula;
1068        //Agafem com data i hora d'entrada la data i hora actuals
1069        DHE.setDataHoraActual();
1070        DHS.resetDataHora(); //perquÈ no tinqui la data hora actuals
1071        //Crear la taula de valors, llegint del fitxer
1072        faula >> nES;
1073        //Es crea la taula dinàmicament, adquireix memòria
1074        ESs = new EntSort [nES];
1075        for(int i=0;i<nES;i++){
1076
1077            faula >> ESs[i].DNI;
```

```
1078                faula >> ESs[i].DHE;
1079                faula >> ESs[i].DHS;
1080            }
1081        }
1082
1083        //DESTRUCTOR
1084        ~Aula()
1085        {
1086            delete [] ESs;
1087        }
1088
1089        //Constructor de copia
1090        Aula(const Aula& AU){
1091
1092            CMAX = AU.CMAX;
1093            nomAula = AU.nomAula;
1094            nES = AU.nES;
1095
1096            ESs = new EntSort [nES];
1097            //Copiem continguts
1098            for (int i = 0; i < nES; i++){
1099                ESs[i] = AU.ESs[i];
1100            }
1101        }
1102
1103
1104         //S'HAN DE REFER JA QUE UNA AULA NO TÉ UNA HORA D'ENTRADA I SORTIDA; TÉ VÀRIES HORES
    D'ENTRADES I SORTIDES, PER CADA ALUMNE
1105        //MËtodes consultors getX
1106
1107        DataHora getDHE()const{
1108
1109            return DHE;
1110        }
1111        DataHora getDHS()const{
1112
1113            return DHS;
1114        }
1115
1116        int getnDNIs(){
1117
1118            return nES;
1119        }
1120
1121        vector<string> getDNIS() const{
1122
1123            vector<string> DNIs;
1124            for (int i = 0; i<nES; i++)
1125            {
1126                DNIs.push_back(ESs[i].DNI);
1127            }
1128                 return DNIs;
1129        }
1130
1131        string getnomAula()const{
1132
1133            return nomAula;
1134        }
1135        int getCMAX()const{
1136
1137            return CMAX;
1138        }
1139        //MËtodes per modificar valors
1140
1141        void setnomAula(string pnomAula){
1142
1143            nomAula = pnomAula;
1144        }
1145
1146        void setCMAX(int pCMAX){
1147
1148            CMAX = pCMAX;
1149        }
1150
1151        void setESs(EntSort* pESs){
1152
1153               ESs = pESs;
1154        }
1155        //Operador  d'asignaciÛ
1156        Aula& operator = (const Aula& AU){ //assigna l'aula a l'objecte
1157
1158            if(this != &AU){ //faig assignaciÛ
1159                //copiar informaciÛ
1160                CMAX = AU.CMAX;
```

```
1161                nomAula = AU.nomAula;
1162                nES = AU.nES;
1163                //Cal destruir la taula i crear-la de nou...
1164                delete [] ESs;
1165                ESs = new EntSort [nES];
1166                for (int i = 0; i < nES; i++){
1167                    ESs[i] = AU.ESs[i];
1168                }
1169            }
1170            return (*this);
1171        }
1172        //Operadors relacionals
1173        bool operator == (Aula& AU){
1174            //comparem component-wise, camp a camp
1175            return (nomAula == AU.nomAula);
1176
1177        }
1178        //Comparem pel nombre d'alumnes que han entrat i sortit
1179        friend bool operator < (Aula& AU1 , Aula&  AU2)
1180        {
1181            return AU1.getnDNIs() < AU2.getnDNIs();
1182        }
1183
1184
1185        // Operadors d'extracciÛ >>
1186
1187        friend istream& operator >> (istream& is, Aula &AU){
1188
1189            is >> AU.CMAX; //llegim capacitat màxima
1190
1191            is >> AU.nES; //llegim el nombre d'entrades i sortides a l'aula
1192
1193
1194            //Adquirim memòria
1195            EntSort* ESsTEMP = new EntSort[AU.nES];
1196
1197            string nomAula, DNIA;
1198            int dde, mme, aaaae;
1199            int hhe, mine, sece;
1200            int dds, mms, aaaas;
1201            int hhs, mins, secs;
1202            int i = 0; //ens en la taula dinàmica
1203            while (is >> nomAula >> DNIA >> dde >> mme >> aaaae >> hhe >> mine >> sece >> dds >>
    mms  >> aaaas >> hhs >> mins >> secs)
1204            {
1205                DataHora DHENT(dde, mme, aaaae, hhe, mine, sece);
1206                DataHora DHSORT(dds, mms, aaaas,hhs,mins,secs);
1207
1208                EntSort EnSr(DNIA,DHENT,DHSORT);
1209
1210                ESsTEMP[i] = EnSr;
1211                i++;
1212            }
1213            AU.setESs(ESsTEMP);
1214
1215        return is;
1216
1217        }
1218        // Operadors d'insercciÛ <<
1219
1220        friend ostream& operator << (ostream& os, Aula &AU){
1221
1222
1223            os << "Fitxa de la Aula: " << endl;
1224            os << "------------------" << endl;
1225            os << "Capacitat mÀxima: " << AU.CMAX << endl;
1226            os << "Nombre de DNIs: " << AU.nES << endl;
1227            os << "Nom Aula: " << AU.nomAula <<endl;
1228            if (AU.nES>0){
1229                for(int i=0; i< AU.nES;i++){
1230                    //Escrivim entrada/sortida i-èssima
1231                 os << AU.ESs[i] << endl;
1232                }
1233            os << endl;
1234            }
1235            os << "------------------" << endl;
1236
1237            return os;
1238        }
1239    };
1240
1241    //Classe Edifici(Llocs)
1242    class Edifici {
1243
```

```
1244        DataHora HEE; //DataHora del horario de entrada
1245        DataHora HES; //DataHora del horario de entrada
1246        string DNIE;
1247        double Temp;
1248
1249    public:
1250        //Constructor per defecte
1251        Edifici(){}
1252        //Constructor amb parametres
1253        Edifici(string pDNIE, DataHora pHEE,DataHora pHES,double pTemp){
1254
1255            HEE = pHEE;
1256            HES = pHES;
1257            DNIE = pDNIE;
1258            Temp = pTemp;
1259        }
1260        Edifici(ifstream& fedifici){
1261
1262            fedifici >> DNIE;
1263            fedifici >> HEE;
1264            fedifici >> HES;
1265            fedifici >> Temp;
1266        }
1267        //Constructor de copia
1268        Edifici(const Edifici& E){
1269
1270            DNIE = E.DNIE;
1271            HEE = E.HEE;
1272            HES = E.HES;
1273            Temp = E.Temp;
1274        }
1275        //MËtodes consultors getX
1276        DataHora getHoraE()const{
1277
1278            return HEE;
1279        }
1280        DataHora getHoraS()const{
1281
1282            return HES;
1283        }
1284        string getDNIE()const{
1285
1286            return DNIE;
1287        }
1288        double getTemp()const{
1289
1290            return Temp;
1291        }
1292        //MËtodes modificadora setX
1293         void setHoraE(DataHora pHEE){
1294
1295            pHEE = HEE;
1296        }
1297        void setHoraS(DataHora pHES){
1298
1299            pHES  = HES;
1300        }
1301        void setDNIE(string pDNIE){
1302
1303            pDNIE = DNIE;
1304        }
1305        void setTemp(double pTemp){
1306
1307            pTemp = Temp;
1308        }
1309        // MetÚdes
1310
1311        void PosibleCOVID(){
1312         list <Persona> Infectats,Sans;
1313            Persona PC,SC;
1314
1315            if(getTemp()>=37.5){
1316                Infectats.push_back(PC);
1317                cout<<"POSSIBLE POSITIU!"<<endl<<"AccÈs denegat"<<endl;
1318            }
1319            else{
1320                Sans.push_back(SC);
1321                cout<<"POT ENTRAR EN EL Edifici!"<<endl<<"AccÈs aceptat"<<endl;
1322            }
1323        }
1324        //Operador  d'asignaciÛ
1325        Edifici& operator = (const Edifici& E){
1326
1327            if(this != &E){
```

```
1328
1329            DNIE = E.DNIE;
1330            HEE = E.HEE;
1331            HES = E.HES;
1332            Temp = E.Temp;
1333
1334            }
1335          return(*this);
1336        }
1337
1338        //Operadors relacionals
1339        friend bool operator == (Edifici& E1, Edifici& E2){
1340
1341            return((E1.DNIE == E2.DNIE) and (E1.HEE == E2.HEE) and (E1.HES == E2.HES) and (E1.Temp
    = E2.Temp));
1342        }
1343
1344        friend bool operator < (Edifici& E1, Edifici& E2){return true;}
1345        friend bool operator > (Edifici& E1, Edifici& E2){return true;}
1346
1347        // Operadors d'extracciÛ >>
1348        friend istream& operator >> (istream& is, Edifici& E){
1349
1350            is >> E.DNIE;
1351            is >> E.HEE;
1352            is >> E.HES;
1353            is >> E.Temp;
1354
1355            return is;
1356        }
1357
1358        friend ostream& operator << (ostream& os, Edifici& E){
1359
1360            os << "Fitxa del Edifici: " << endl;
1361            os << "-------------------" << endl;
1362            os << "DNI: " << E.DNIE << endl;
1363            os << E.HEE << endl;
1364            os << E.HES << endl;
1365            os << "Temperatura: " << E.Temp << endl;
1366            os << "-------------------" << endl;
1367
1368            return os;
1369        }
1370
1371    };
1372
1373    //Classe Professor
1374    class Professor : public Persona{
1375
1376    //-------------->Dades<--------------//
1377        DataHora HPE,HPS;
1378        int Curs;
1379        string Aula,Materia,DNIPP;
1380
1381    public:
1382        //-------------->API<--------------//
1383
1384        //----------->Constructors<----------//
1385
1386        //Constructor per defecte
1387        Professor(){}
1388        //Constructor amb parametres
1389        Professor(Persona P, string pDNIPP,string pAula,string pMateria,DataHora pHPE, DataHora
    pHPS):Persona(P){
1390
1391            DNIPP = pDNIPP;
1392            Aula = pAula;
1393            Materia = pMateria;
1394            HPE = pHPE;
1395            HPS = pHPS;
1396
1397
1398        }
1399        //Constructor amb parametres des d'un fitxer
1400        Professor(Persona P,ifstream& fprofessors):Persona(P){
1401
1402            fprofessors >> DNIPP;
1403            fprofessors >> Aula;
1404            fprofessors >> Materia;
1405            fprofessors >> HPE;
1406            fprofessors >> HPS;
1407
1408        }
1409        //Constructor de copia
```

```
1410        Professor(const Professor& PP):Persona(PP){
1411
1412            DNIPP = PP.DNIPP;
1413            Aula = PP.Aula;
1414            Materia = PP.Materia;
1415            HPE = PP.HPE;
1416            HPS = PP.HPS;
1417        }
1418        //Mètodes consultors getX
1419        string getDNIPP()const {
1420
1421            return DNIPP;
1422        }
1423        string getAula()const{
1424
1425            return Aula;
1426        }
1427        string getMateria()const{
1428
1429            return Materia;
1430        }
1431        DataHora getHoraHPE()const{
1432
1433            return HPE;
1434        }
1435        DataHora getHoraHPS()const{
1436
1437            return HPS;
1438        }
1439        //MËtodes modificadors setX
1440        void setDNIPP (string pDNIPP) {
1441
1442            pDNIPP = DNIPP;
1443        }
1444        void setAula(string pAula){
1445
1446            pAula = Aula;
1447        }
1448        void setMateria(string pMateria){
1449
1450            pMateria = Materia;
1451        }
1452        void setHoraHPE(DataHora pHPE){
1453
1454            pHPE = HPE;
1455        }
1456        void setHoraHPS(DataHora pHPS){
1457
1458            pHPS = HPS;
1459        }
1460        //Operador  d'asignació
1461        Professor& operator = (const Professor& PP){
1462
1463            if(this !=&PP){
1464
1465            Persona::operator = (PP);
1466            DNIPP = PP.DNIPP;
1467            Aula = PP.Aula;
1468            Materia = PP.Materia;
1469            HPE = PP.HPE;
1470            HPS = PP.HPS;
1471
1472            }
1473            return (*this);
1474        }
1475        //Operadors relacionals
1476        friend bool operator == (Professor& PP1 , Professor&  PP2){
1477
1478            return((PP1.DNIPP == PP2.DNIPP) and (PP1.Aula == PP2.Aula) and (PP1.Materia ==
    PP2.Materia)and (PP1.HPE == PP2.HPE) and (PP1.HPS == PP2.HPS));
1479
1480        }
1481        friend bool operator < (Professor& PP1 , Professor&  PP2){return true;}
1482        friend bool operator > (Professor& PP1 , Professor&  PP2){return true;}
1483
1484        // Operadors d'extracciÔ >>
1485
1486        friend istream& operator >> (istream& is, Professor& PP){
1487
1488            Persona P; // La part Persona de A
1489            //is >> P;
1490            DataHora HPE;
1491            DataHora HPS;
1492            string Aula,Materia,DNIPP;
```

```
1493
1494            is >> DNIPP;
1495            is >> Aula;
1496            is >> Materia;
1497            is >> HPE;
1498            is >> HPS;
1499
1500            Professor PR (P,DNIPP,Aula,Materia,HPE,HPS);
1501            PP = PR;
1502
1503            return is;
1504        }
1505        // Operadors d'inserció <<
1506
1507        friend ostream& operator << (ostream& os, Professor& PP){
1508
1509            Persona P = (Persona) PP; //Extrau la part de la persona de A
1510            os << "Fitxa del Professor/a: " << endl;
1511            os << "------------------" << endl;
1512            //os << P;
1513            os << "DNI: " << PP.DNIPP << endl;
1514            os << "Aula: " << PP.Aula << endl;
1515            os << "Materia: " << PP.Materia << endl;
1516            os << "Hora d'entrada: " << PP.HPE << endl;
1517            os << "Hora de sortida: " << PP.HPS << endl;
1518            os << "------------------" << endl;
1519
1520            return os;
1521        }
1522    };
1523
1524    //Classe PersonalExtern
1525    class PersonalExtern : public Persona{
1526    //-------------->Dades<-------------//
1527        DataHora HPE; //DataHora de PE Entrada
1528        DataHora HPS; //DataHora de PE Sortida
1529        string Aula;
1530        string Ocupacio;
1531        string DNIPE;
1532    public:
1533        //-------------->API<-------------//
1534
1535        //----------->Constructors<----------//
1536
1537        //Constructor per defecte
1538        PersonalExtern(){}
1539        //Constructor amb parametres
1540
1541        PersonalExtern(Persona P,string pDNIPE,string pAula,string pOcupacio,DataHora pHPE,
    DataHora pHPS):Persona(P){
1542
1543            HPS = pHPS;
1544            HPE = pHPE;
1545            Aula = pAula;
1546            Ocupacio = pOcupacio;
1547            DNIPE = pDNIPE;
1548
1549        }
1550        //Constructor amb parametres des d'un fitxer
1551        PersonalExtern(Persona P,ifstream& fpersonalextern):Persona(P){
1552
1553            fpersonalextern >> DNIPE;
1554            fpersonalextern >> Aula;
1555            fpersonalextern >> Ocupacio;
1556            fpersonalextern >> HPE;
1557            fpersonalextern >> HPS;
1558        }
1559        //Constructor de copia
1560        PersonalExtern(const PersonalExtern& PE):Persona(PE){
1561
1562            DNIPE = PE.DNIPE;
1563            Aula = PE.Aula;
1564            Ocupacio = PE.Ocupacio;
1565            HPE = PE.HPE;
1566            HPS = PE.HPS;
1567
1568        }
1569        //MËtodes consultors getX
1570        DataHora getHPE()const{
1571
1572            return HPE;
1573        }
1574        DataHora getHPS()const{
1575
```

```
1576            return HPS;
1577        }
1578        string getAula()const{
1579
1580            return Aula;
1581        }
1582        string getOcupacio()const{
1583
1584            return Ocupacio;
1585        }
1586        string getDNIPE()const{
1587
1588            return DNIPE;
1589        }
1590        //MËtodes modificadors setX
1591        void setHPE(DataHora pHPE){
1592
1593            pHPE = HPE;
1594        }
1595        void setHPS(DataHora pHPS){
1596
1597            pHPS = HPS;
1598        }
1599        void setAula(string pAula){
1600
1601            pAula = Aula;
1602        }
1603        void setOcupacio(string pOcupacio){
1604
1605            pOcupacio = Ocupacio;
1606        }
1607        void setDNIPE(string pDNIPE){
1608
1609            pDNIPE = DNIPE;
1610        }
1611        //Operador  d'asignaciÛ
1612        PersonalExtern& operator = (const PersonalExtern& PE){
1613
1614            if(this !=&PE){
1615
1616            Persona::operator = (PE);
1617            DNIPE = PE.DNIPE;
1618            Aula = PE.Aula;
1619            Ocupacio = PE.Ocupacio;
1620            HPE = PE.HPE;
1621            HPS = PE.HPS;
1622
1623            }
1624            return (*this);
1625        }
1626        //Operadors relacionals
1627        friend bool operator == (PersonalExtern& PE1 , PersonalExtern&  PE2){
1628
1629            return((PE1.DNIPE==PE2.DNIPE) and (PE1.Aula == PE2.Aula) and (PE1.Ocupacio ==
    PE2.Ocupacio) and (PE1.HPE==PE2.HPE)and (PE1.HPS==PE2.HPS));// and (PE1.h==PE2.h)
    and(PE1.m==PE2.m) and (PE1.s==PE2.s));
1630
1631        }
1632        friend bool operator < (PersonalExtern& PE1 , PersonalExtern&  PE2){return true;}
1633        friend bool operator > (PersonalExtern& PE1 , PersonalExtern&  PE2){return true;}
1634
1635        // Operadors d'extracciÛ >>
1636
1637        friend istream& operator >> (istream& is, PersonalExtern &PE){
1638
1639            Persona P; // La part Persona de A
1640            //is >> P;
1641            DataHora HPE;
1642            DataHora HPS;
1643            string Aula,Ocupacio,DNIPE;
1644            is >> DNIPE;
1645            is >> Aula;
1646            is >> Ocupacio;
1647            is >> HPE;
1648            is >> HPS;
1649            PersonalExtern PR (P,DNIPE,Aula,Ocupacio,HPE,HPS);
1650            PE = PR;
1651
1652            return is;
1653        }
1654        // Operadors d'inserciÛ <<
1655
1656        friend ostream& operator << (ostream& os, PersonalExtern& PE){
1657
```

```
1658            Persona P = (Persona) PE; //Extrau la part de la persona de A
1659            os << "Fitxa de la Personal Extern: " << endl;
1660            os << "------------------" << endl;
1661            //os << P;
1662            os << "DNI: " << PE.DNIPE << endl;
1663            os << "Aula: " << PE.Aula << endl;
1664            os << "OcupaciÛ/C‡rrec: " << PE.Ocupacio << endl << endl ;
1665            os << "Entrada: " <<endl;
1666            os << PE.HPE;
1667            os << endl ;
1668            os << "Sortida: " <<endl;
1669            os << PE.HPS;
1670            os << "------------------" << endl;
1671
1672            return os;
1673        }
1674    };
1675
1676    //Menu Persona
1677    void MenuPersona (int& opP){
1678        setlocale(LC_ALL,"Catalan");
1679        cout <<" ----------------------------------------" << endl;
1680        cout <<" Escull una opcio. " << endl;
1681        cout << "1. Desplegar la llista amb la informacio de cada persona que entra a l'edifici "
    << endl;
1682        cout << "2. Afegir positiu al registre " << endl;
1683        cout << "3. Eliminar positiu al registre" << endl;
1684        cout << "4. Coneixer la quantitat maxima de persones que hi ha dins de l'edifici " << endl;
1685        cout << "5. Acabar" << endl;
1686        cout <<" ----------------------------------------" << endl;
1687        cin >> opP;
1688        while(opP < 1 or opP > 5){
1689            cout << "Opcio incorrecte. Torna a escollir una opcio:" << endl;
1690            cin >> opP;}
1691    }
1692
1693    //Menu Professor
1694    void MenuProfessor(int& opS){
1695
1696        setlocale(LC_ALL,"Catalan");
1697        cout <<" ----------------------------------------" << endl;
1698        cout <<" Escull una opcio. " << endl;
1699        cout << "1. Desplegar la llista del professors dins l'edifici " << endl;
1700        cout << "2. Coneixer la quantitat de professors assignats en cada aula " << endl;
1701        cout << "3. Coneixer la quantitat de professors dins d'edifici" << endl;
1702        cout << "4. Acabar" << endl;
1703        cout <<" ----------------------------------------" << endl;
1704        cin >> opS;
1705        while(opS< 1 or opS > 4){
1706           cout << "Opcio incorrecte. Torna a escollir una opcio:" << endl;
1707            cin >> opS;
1708        }
1709    }
1710
1711    //Menu Alumne
1712    void MenuAlumne(int& opA){
1713
1714        setlocale(LC_ALL,"Catalan");
1715        cout <<" ----------------------------------------" << endl;
1716        cout <<" Escull una opcio. " << endl;
1717        cout << "1. Desplegar la lista d'alumnes que han entrat a l'edifici " << endl;
1718        cout << "2. Afegir positiu al registre" << endl;
1719        cout << "3. Eliminar positiu al registre " << endl;
1720        cout << "4. Coneixer la quantitat màxima de persones que hi ha dins de l'edifici " << endl;
1721        cout << "5. Buscar a l'alumne mitjaçant el DNI" << endl;
1722        cout << "6. Acabar" << endl;
1723        cout <<" ----------------------------------------" << endl;
1724        cin >> opA;
1725        while(opA < 1 or opA > 6){
1726            cout << "Opcio incorrecte. Torna a escollir una opcio:" << endl;
1727            cin >> opA;
1728        }
1729    }
1730
1731    //Menu Edifici
1732    void MenuEdifici(int& opE){
1733
1734        setlocale(LC_ALL,"Catalan");
1735        cout <<" ----------------------------------------" << endl;
1736        cout <<" Escull una opció. " << endl;
1737        cout << "1. Desplegar la llista de possibles contagiats " << endl;
1738        cout << "2. Desplegar la llista dels no contagiats " << endl;
1739        cout << "3. Entrar persona a l'edifici " << endl;
1740        cout << "4. Sortir persona de l'edifici" << endl;
```

```
1741        cout << "5. Confinar Aula de l'edifici" << endl;
1742        cout << "6. Acabar" << endl;
1743        cout <<" ----------------------------------------" << endl;
1744        cin >> opE;
1745        while(opE < 1 or opE > 6){
1746            cout << "Opcio incorrecte. Torna a escollir una opcio:" << endl;
1747            cin >> opE;
1748        }
1749    }
1750
1751    //Menu Aula
1752    void MenuAula(int& opAU){
1753
1754        setlocale(LC_ALL,"Catalan");
1755        cout <<" ----------------------------------------" << endl;
1756        cout <<" Escull una opció. " << endl;
1757        cout << "1. " << endl;
1758        cout << "2. " << endl;
1759        cout << "3. " << endl;
1760        cout << "4. " << endl;
1761        cout << "5. " << endl;
1762        cout << "6. Acabar" << endl;
1763        cout <<" ----------------------------------------" << endl;
1764        cin >> opAU;
1765        while(opAU < 1 or opAU > 6){
1766            cout << "Opcio incorrecte. Torna a escollir una opcio:" << endl;
1767            cin >> opAU;
1768        }
1769    }
1770
1771    //Menu Personal Extern
1772    void MenuPersonaExtern(int& opPE){
1773
1774        setlocale(LC_ALL,"Catalan");
1775        cout <<" ----------------------------------------" << endl;
1776        cout <<" Escull una opcio. " << endl;
1777        cout << "1. Desplegar la llista del personal extern dins l'edifici " << endl;
1778        cout << "2. Conèixer la quantitat de personal extern assignats en cada aula " << endl;
1779        cout << "3. Conèixer la quantitat de personal extern dins d'edifici" << endl;
1780        cout << "4. Acabar" << endl;
1781        cout <<" ----------------------------------------" << endl;
1782        cin >> opPE;
1783        while(opPE < 1 or opPE > 4){
1784            cout << "Opcio incorrecte. Torna a escollir una opcio:" << endl;
1785            cin >> opPE;
1786        }
1787
1788    }
1789
1790    //Menu Principal
1791    void MenuPrincipal(int &option){
1792
1793            setlocale(LC_ALL,"Catalan");
1794            cout <<"
    ----------------------------------------------------------------------------------------------"
    << endl;
1795            cout << "Benvingut al Menu Principal del programa sobre la Gestio d'Aules Covid-19 d'un
    Centre Docent "<<endl;
1796            cout << endl << endl;
1797            cout << "MENU D'OPCIONS: Escull una opcio." << endl;
1798            cout << "\n1. Menu Persona" << endl;
1799            cout << "\n2. Menu Aula" << endl;
1800            cout << "\n3. Menu Alumne" << endl;
1801            cout << "\n4. Menu Professor " << endl;
1802            cout << "\n5. Menu Personal Extern" << endl;
1803            cout << "\n6. Menu Edifici" << endl;
1804            cout << "\n7. Finalitzar!" << endl;
1805            cout <<"
    ----------------------------------------------------------------------------------------------"
    << endl;
1806            cin >> option;
1807            while (option < 1 || option > 7){
1808                cout << "OpciÛ incorrecte. Torna a escollir una opciÛ:" << endl;
1809                cin >> option;
1810            }
1811    }
1812
1813    //VOIDS.......
1814    /**
1815    - Eliminar posibles contagis alhora de l'entrada a l'edifici HECHO (LIST)
1816    - Poder gestionar les entrades i sortides del centre, inclÚs líhora del pati. HECHO (QUEUE)
1817    - Capacitat de poder confinar una aula si se'n registren mËs de tres
1818      positius en aquesta. HECHO KIRIAN EL CRACK
1819
```

```
- ConËixer la quantitat de grups (persona ,alumnes, professors, personal
  extern) que hi ha dins l'edifici i calcular el total. STACK HECHO

- Conèixer la quantitat d'alumnes que hi ha en una aula segons el
  moment del dia. HECHO  EN CADA CLASE

- Presenciar la distribuciÛ de seguretat díaules i dels edificis respecte a la
  relaciÛ espai/persones.

- ¡¡¡ Cálculos estadisticos !!!

- Tenir la capacitat d'afegir i eliminar positius al registre.

**/
// Menu Interactiu
int main(void){

    setlocale(LC_ALL,"Catalan");
    int option;
    MenuPrincipal(option);
    while (1 <= option && option <= 6){
        switch (option){
            case 1: {

                setlocale(LC_ALL,"Catalan");
                cout << "Has escollit el Menu de Persona" << endl;
                cout << "\nBenvingut al Menu de Persona " << endl;
                ifstream fpersona("personas.txt");
                Persona P(fpersona);
                stack<Persona,vector<Persona> > GestioPersona;
                int opP;
                MenuPersona(opP);
                while (1 <= opP & opP <= 4){
                        switch (opP){
                        case 1: {
                                cout << "1.Desplegar la llista amb la informacio de cada persona que entra a l'edifici" <<endl;
                                vector<Persona> vP;
                                vP.push_back(P);
                                while(fpersona>>P){

                                vP.push_back(P);
                                }
                                for(int i=0;i<vP.size();i++){

                                cout << vP[i] << endl;
                                }
                                break;
                        }
                        case 2: {
                                cout << "2.Afegir positiu al registre " <<endl;
                                //queue<Persona> GestioPersona;
                                string NOM,DNI,COGNOM;int Edat;char Sexe;
                                cout << "Introdueix un nom: "<<endl; cin >> NOM;
                                cout << "Introdueix un Cognom: "<<endl; cin >>COGNOM;
                                cout << "Introdueix un DNI: "<<endl; cin >> DNI;
                                cout << "Introdueix un Edat: "<<endl; cin >> Edat;
                                cout << "Introdueix un Sexe(D/H ,Dona = D , Home = H) : "<<endl; cin >> Sexe;
                                Persona P(NOM,COGNOM,DNI,Edat,Sexe);
                                GestioPersona.push(P);
                                cout << P;
                                break;
                        }
                        case 3:{
                                cout << "3.Eliminar positiu al registre" <<endl;
                                //queue<Persona>GestioPersona;
                                if(!GestioPersona.empty()){
                                P = GestioPersona.top();
                                cout << "Nom : " << P.getNom()<< " Cognom: " << P.getCognom() << " DNI: " << P.getDNI() << " Edat:  " << P.getEdat() << " Sexe(Home H/ Dona D ):  " << P.getSexe() << endl;
                                GestioPersona.pop();
                        }
                        else {cout << "No hi ha persones dins l'edifici!"<< endl;}
                                break;
                        }
                        case 4:{
                                cout << "Coneixer la quantitat màxima de persones que hi ha dins de l'edifici" <<endl;
                                Persona PE;
                                stack <Persona> personas;
                                personas.push(PE);
                                int cont=0;
```

```
1899                                    while(fpersona>>PE){
1900                                        personas.push(PE);
1901                                    }
1902                                    cont = personas.size();
1903                                    cout << " Total de persones dins l'edifici : "<< cont << endl;
1904                                    break;
1905                                }
1906                                case 5: {
1907                                    cout << "Buscar a l'alumne mitjacant el DNI" << endl;
1908                                    map<string,Persona> BuscaPersona;
1909                                    map<string,Persona>:: iterator it;
1910                                    string dni,cog,nom;int edat;char sexe;
1911                                    Persona P;
1912                                    while(fpersona>>P){
1913                                    Persona P(nom,cog,dni,edat,sexe);
1914                                    BuscaPersona.insert(dni,P);
1915                                    }
1916                                    /*
1917                                    cout << "Introdueixi el DNI " << endl; cin >> DNI;
1918                                    for(it = BuscaPersona.begin(); it != BuscaPersona.end();it++){
1919
1920                                        it
1921                                    */
1922
1923                                    break;
1924                                }
1925                                default: {}
1926                                }
1927                            MenuPersona(opP);
1928                            }
1929                        MenuPrincipal(option);
1930                     break;
1931                    }
1932                    case 2: {
1933                        cout << "Has escollit el Menu Aules" << endl;
1934                        cout << "\nBenvingut al Menu d'aules " << endl;
1935                        ifstream faula1("TR1.txt");
1936                        Aula A1(faula1);
1937                        cout << A1;
1938                        ifstream faula2("TR2.txt");
1939                        Aula A2(faula2);
1940                        cout << A2;
1941                        ifstream faula3("TR3.txt");
1942                        Aula A3(faula3);
1943                        cout << A3;
1944                        /*vector<Aula> vAU;
1945                        vAU.push_back(AU);
1946                        while(faula>>AU){
1947                           vAU.push_back(AU);
1948                        }
1949                        for(int i=0;i<vAU.size();i++){
1950                            cout << vAU[i] << endl;
1951                        }
1952                        */
1953                        break;
1954                    }
1955                    case 3: {
1956
1957                        setlocale(LC_ALL,"Catalan");
1958                        cout << "Has escollit el Menu Alumnes" << endl;
1959                        cout << "\nBenvingut al Menu d'alumnes " << endl;
1960                        ifstream falumne("alumnes.txt");
1961                        Persona P;
1962                        Alumne AL(P,falumne);
1963                        int opA;
1964                        MenuAlumne(opA);
1965                        while (1 <= opA & opA <= 3){
1966                                switch (opA){
1967                                    case 1: {
1968                                        cout << "1.Desplagar la llista d'alumnes que han entrat a
    l'edifici " <<endl;
1969                                        vector<Alumne> vA;
1970                                        vA.push_back(AL);
1971                                        while(falumne>>AL){
1972                                            vA.push_back(AL);
1973                                        }
1974                                        for(int i=0;i<vA.size();i++){
1975                                            cout << vA[i] << endl;
1976                                            }
1977                                        break;
1978                                    }
1979                                    case 2: {
1980                                        cout << "2.Coneixer la quantitat d'alumnes que hi ha en cada
    aula" <<endl;
```

```
1981                                    vector<Alumne> vA;
1982                                    vA.push_back(AL);
1983                                    int cont1=1,cont2=0,cont3=0;
1984                                    while(falumne>>AL){
1985                                        vA.push_back(AL);
1986                                        if(AL.getAula()=="TR1-001"){cont1++;}
1987                                        else if(AL.getAula()=="TR2-001"){cont2++;}
1988                                        else if(AL.getAula()=="TR3-001"){cont3++;}
1989
1990                                    }
1991                                    cout<<"Total d'alumnes en la classe TR1-001 es de : "<< cont1
    << endl;
1992                                    cout<<"Total d'alumnes en la classe TR2-001 es de : "<< cont2
    << endl;
1993                                    cout<<"Total d'alumnes en la classe TR3-001 es de : "<< cont3
    << endl;
1994                                    break;
1995                                }
1996                                case 3: {
1997                                    cout << "3.Coneixer la quantitat d'alumnes en un edifici" <<endl;
1998                                    vector<Alumne> vA;
1999                                    int contA=0;
2000                                    vA.push_back(AL);
2001                                    while(falumne>>AL){
2002                                        vA.push_back(AL);
2003
2004                                    }
2005                                    contA=vA.size();
2006                                    cout << "Total d'alumnes dins l'edifici : " << contA << endl;
2007                                    break;
2008                                }
2009                                default: {}
2010                            }
2011                        MenuAlumne(opA);
2012                        }
2013                    MenuPrincipal(option);
2014                break;
2015                }
2016                case 4: {
2017                    cout << "Has escollit el Menu de Professors" << endl;
2018                    cout << "\nBenvingut al Menu de professors " << endl;
2019                    ifstream fprofessors("professor.txt");
2020                    Persona P;
2021                    Professor PP(P,fprofessors);
2022                    int opS;
2023                    MenuProfessor(opS);
2024                    while (1 <= opS & opS <= 3){
2025                            switch (opS){
2026                                case 1: {
2027                                    cout << "1.Desplegar la llista del professors dins l'edifici "
    <<endl;
2028                                    vector<Professor> vPP;
2029                                    vPP.push_back(PP);
2030                                    while(fprofessors>>PP){
2031
2032                                        vPP.push_back(PP);
2033                                    }
2034                                    for(int i=0;i<vPP.size();i++){
2035
2036                                        cout << vPP[i] << endl;
2037                                    }
2038                                    break;
2039                                }
2040                                case 2: {
2041                                    cout << "2.Coneixer la quantitat de professors assignats en
    cada aula " <<endl;
2042                                    vector<Professor> vPP;
2043                                    vPP.push_back(PP);
2044                                    int cont1=1,cont2=0,cont3=0;
2045                                    while(fprofessors>>PP){
2046                                        vPP.push_back(PP);
2047
2048                                        if(PP.getAula()=="TR1-001"){cont1++;}
2049                                        else if(PP.getAula()=="TR2-001"){cont2++;}
2050                                        else if(PP.getAula()=="TR3-001"){cont3++;}
2051                                    }
2052                                    cout<<"Total de professors en la classe TR1-001 es de : "<<
    cont1 << endl;
2053                                    cout<<"Total de professors en la classe TR2-001 es de : "<<
    cont2 << endl;
2054                                    cout<<"Total de professors en la classe TR3-001 es de : "<<
    cont3 << endl;
2055                                    break;
2056                                }
```

```
                                case 3: {
                                    cout << "3. Coneixer la quantitat de professors dins d'edifici"
<<endl;
                                    vector<Professor> vPP;
                                    int contP=0;
                                    vPP.push_back(PP);
                                    while(fprofessors>>PP){

                                        vPP.push_back(PP);
                                    }
                                    contP=vPP.size();
                                    cout << "Total de professors  dins l'edifici : " << contP <<
endl;
                                    break;
                                }
                                default: {}
                            }
                        MenuProfessor(opS);
                        }
                    MenuPrincipal(option);
                 break;
                }
                case 5: {
                    setlocale(LC_ALL,"Catalan");
                    cout << "Has escollit el Menu Personal Extern " << endl;
                    cout << "\nBenvingut al Menu del personal Extern " << endl;
                    ifstream fpersonalextern("personalextern.txt");
                    Persona P;
                    PersonalExtern PE(P,fpersonalextern);
                    int opPE;
                    MenuAlumne(opPE);
                    while (1 <= opPE & opPE <= 3){
                            switch (opPE){
                                case 1: {
                                    cout << "1.Coneixer la quantitat de personal extern assignats
en cada aula " <<endl;
                                    vector<PersonalExtern> vPE;
                                    vPE.push_back(PE);
                                    while(fpersonalextern>>PE){
                                        vPE.push_back(PE);
                                    }
                                    for(int i=0;i<vPE.size();i++){
                                        cout << vPE[i] << endl;
                                        }
                                    break;
                                }
                                case 2: {
                                    cout << "2.Coneixer la quantitat de personal extern assignats
en cada aula" <<endl;
                                    vector<PersonalExtern> vPE;
                                    vPE.push_back(PE);
                                    int cont1=1,cont2=0,cont3=0;
                                    while(fpersonalextern>>PE){
                                        vPE.push_back(PE);
                                        if(PE.getAula()=="TR1-001"){cont1++;}
                                        else if(PE.getAula()=="TR2-001"){cont2++;}
                                        else if(PE.getAula()=="TR3-001"){cont3++;}

                                    }
                                    cout<<"Total d'alumnes en la classe TR1-001 es de : "<< cont1
<< endl;
                                    cout<<"Total d'alumnes en la classe TR2-001 es de : "<< cont2
<< endl;
                                    cout<<"Total d'alumnes en la classe TR3-001 es de : "<< cont3
<< endl;
                                    break;
                                }
                                case 3: {
                                    cout << "3.Coneixer la quantitat de personal extern dins
d'edifici" <<endl;
                                    vector<PersonalExtern> vPE;
                                    int contPE=0;
                                    vPE.push_back(PE);
                                    while(fpersonalextern>>PE){
                                        vPE.push_back(PE);

                                    }
                                    contPE=vPE.size();
                                    cout << "Total d'alumnes dins l'edifici : " << contPE << endl;
                                    break;
                                }
                                default: {}
                            }
                        MenuPersonaExtern(opPE);
```

```
                    }
                MenuPrincipal(option);
             break;
            }
            case 6: {
                cout << "Has escollit el Menu Edifici" << endl;
                cout << "\nBenvingut al Menu d'edificis " << endl;
                ifstream fedifici("edificis.txt");
                Edifici EE(fedifici);

                queue<Edifici> GestioEdifici;
                int opE;
                MenuEdifici(opE);
                while (1 <= opE & opE <= 5){
                    switch (opE){

                    case 1:{
                     cout << "1.Desplegar la llista de possibles contagiats" <<endl;
                        list<Edifici> Infectats;
                        list<Edifici>:: iterator it = Infectats.begin();
                        string dnie; DataHora E,S;double temp;
                        while(fedifici >> dnie >> E >> S >>temp){
                            if(temp>=37.5){
                                    Edifici EI(dnie,E,S,temp);
                                    Infectats.push_back(EI);

                            }
                        }

                      for(it=Infectats.begin();it!=Infectats.end();it++){
                        cout<<"----------------------------"<<endl;
                        cout<<*it<<endl;
                        cout<<"Possible infectat, acces denegat!"<<endl;
                        cout<<"----------------------------"<<endl;
                        }
                    break;
                    }
                    case 2:{
                        cout << "2.Desplagar la llista dels no contagiats" <<endl; //NO
FUNCIONA SI USAS EL CASE 1, HAY QUE SALIR AL MENU PRINCIPAL "5" Y VOLVER A ESTE CASE
                        ///Eliminar posibles contagis alhora de l'entrada a l'edifici
                        list<Edifici> Sans;
                        list<Edifici>:: iterator it2 = Sans.begin();
                        string dnie2; DataHora E2,S2;double temp2;
                        while(fedifici >> dnie2 >> E2 >> S2 >>temp2){
                            if(temp2<37.5){
                                    Edifici ES(dnie2,E2,S2,temp2);
                                    Sans.push_back(ES);
                                }
                        }
                        for(it2=Sans.begin();it2!=Sans.end();it2++){
                            cout<<"----------------------------"<<endl;
                            cout<<*it2<<endl;
                            cout<<"Pots entrar."<<endl;
                            cout<<"----------------------------"<<endl;
                        }
                    break;
                    }
                    case 3:{
                        cout << "3.Entrar persona a l'edifici" <<endl;
                        string DNIE; double Temp;
                        DataHora HEE; //DataHora d'entrada
                        DataHora HES; //DataHora de sortida

                        cout << "Introdueix un DNI: "<<endl; cin >> DNIE;
                        cout << "Introdueix una DataHora d'entrada(5/06/2021 19:25:35 nomes els
numeros)  " << endl; cin >> HEE;
                        cout << "Introdueix una DataHora de sortida(6/06/2021 20:20:15 nomes
els numeros)  " << endl; cin >> HES;
                        cout << "Introdueix una Temperatura entre 35.0 i 41 : "<<endl; cin >>
Temp;
                        Edifici EE(DNIE,HEE,HES,Temp);
                        GestioEdifici.push(EE);
                        cout<<EE;
                        break;
                    }
                    case 4: {
                        cout << "4.Sortir persona de l'edifici" <<endl;
                        if(!GestioEdifici.empty()){
                            Edifici EE = GestioEdifici.front();
                            cout << "Esta sortint la persona amb DNI: "<<EE.getDNIE()<<" i amb
una temperatura de "<<EE.getTemp()<<endl;
                            GestioEdifici.pop();
                        }
```

```
                        else {cout << "No hi ha persones dins l'edifici!"<< endl;}
                    break;
                    }
                    case 5:{
                        cout << "5. Confinar Aula de l'edifici" << endl;
                        vector<Edifici> Total;
                        vector<Edifici> Infectats2;
                        list<Edifici> Infectats;
                        int cont=0;
                        string dniC; DataHora EC,SC;double tempC;
                        while(fedifici >> dniC >> EC >> SC >>tempC){
                            Edifici D(dniC,EC,SC,tempC);
                            Total.push_back(D);
                            if(tempC>=37.5){
                                    Edifici EI(dniC,EC,SC,tempC);
                                    Infectats2.push_back(EI);
                            }
                        }
                        for(int i=0;i<=Infectats2.size();i++){
                            for(int y=0;y<=Total.size();y++){
                                    if(Infectats2[i].getDNIE()==Total[y].getDNIE()){
                                        cont++;
                                    }
                            }
                        }
                        if(cont>=5){
                            cout<<endl<<endl<<"WARNING  WARNING WARNING  WARNING WARNING  WARNING WARNING  WARNING "<<endl<<endl;
                            cout<<"L'edifici ha sigut confinat per excÃ©s de positius de COVID-19"<<endl<<endl;
                            cout<<"WARNING  WARNING WARNING  WARNING WARNING  WARNING WARNING  WARNING "<<endl<<endl;
                        }
                        else cout<<"Edifici sense suficients contagis"<<endl;

                        break;
                    }
                default: {}
                }
            MenuEdifici(opE);
            }
          }
        }
    MenuPrincipal(option);
    }
cout << "\nPrograma finalitzat fins aviat! " << endl;
}
```