Corrigé de l'examen de rattrapage

Exercice 1 :

```
typedef struct {int jour,mois,annee; } date;
typedef date typelem;
typedef struct { typelem t[max];
                 int sommet;
               } pile;
```

void permuter(date *A, date *B)

```
{ date X;
  X=*A;  *A=*B; *B=X;
}
```

int comparedate(date A, date B)

```
{ if (A.annee>B.annee) return -1;
  else if (A.annee<B.annee) return 1;
      else if (A.mois>B.mois) return -1;
          else if (A.mois<B.mois) return 1;
              else if (A.jour>B.jour) return -1;
                  else if (A.jour<B.jour) return 1;
                      else return 0;
}
```

void copiepile(pile *p1,pile *p2)

```
{ typelem x;
  while(!pilevide(*p2)){desempiler(p2,&x);empiler(p1,x);}
}
```

```
void triPile(pile *p)
{ pile r=initpile(), s=initpile();     typelem min,x; printf("TRI\n");
  while(!pilevide(*p))
  {  desempiler(p,&min);
     while(!pilevide(*p))
     {   desempiler(p,&x);
         if (comparedate(x,min)==1) permuter(&x,&min);
         empiler(&r,x);
     }
     empiler(&s,min);
     copiepile(p,&r);
  } copiepile(p,&s);
}
```

Exercice 2 :

```
typedef struct {int min,max;} interval;

interval creerInterval(int min, int max)
{ interval I;
  I.min=min;
  I.max=max;
  return I;
}
int intervalVide(interval I)
{  if (I.min>I.max) return 1;
   else return 0;
}
int cardinalite(interval I)
{ if (intervalVide(I)==1) return 0;
   else return(I.max-I.min+1);
}
```

int appartient (int x, interval I)

```
{ if (intervalVide(I)==1) return 0;
  else if (x>=I.min && x<=I.max) return 1;
       else return 0;
}
```

int inclus(interval I1, interval I2)

```
{  if ( intervalVide(I1) || intervalVide(I2) ) return 0;
   else if (appartient(I1.min,I2) && appartient(I1.max,I2)) return 1;
}
```

int memeInterval(interval I1, interval I2)

```
{ if (I1.min==I2.min && I1.max==I2.max) return 1;
  else return 0;
}
```

interval intersection (interval I1, interval I2)

```
{ interval I3;
  if (memeInterval(I1,I2) || inclus(I1,I2)) return I1;
  if (inclus(I2,I1)) return I2;
  if (I1.min>=I2.min) I3.min=I1.min;
                else I3.min=I2.min;
  if (I2.max<=I1.max) I3.max=I2.max;
                else I3.max=I1.max;
  return(I3);
}
```

int chevauche (interval I1, interval I2)

```
{  if (!intervalVide(intersection(I1,I2))) return 1;
  else return 0;
}
```

interval unionI(interval I1, interval I2)

```
{ interval I3;
  if (I1.min<=I2.min) I3.min=I1.min; else I3.min=I2.min;
  if (I1.max>=I2.max) I3.max=I1.max; else I3.max=I2.max;
  return I3;
}

typedef struct ne *ensemble;
typedef struct ne {interval elt;
                   ensemble svt; } noeud;
```

void inserer (ensemble *L, interval I)

```
{ ensemble p=*L, prd, R=creernoeud() ; R->elt=I ;
  while (p!=NULL && p->elt.min<I.min)
   {prd=p; p=p->svt ; }
   if (p==*L) {R->svt=*L; *L=R; }
   else  {prd->svt=R; R->svt=p ; }
}
```

void supprimer (ensemble *L, interval I)

```
{ ensemble p=*L, prd;
  while (p!=NULL && !memeInterval(p->elt, I) )
   {prd=p; p=p->svt ; }
    if (p!=NULL)
      { if (p==*L)  *L=*L->svt;
        else  prd->svt=p->svt;
        free(p);
       }
}
```

ensemble sousEnsemble(ensemble L, interval I)

```
{
}
```

/* QUESTION NON RETENUE */

interval fusion (ensemble SL, interval I)

{

}