Le 03/02/11

EXAMEN DE RATTRAPAGE (durée 1h30)

Exercice 1 (7 pts) : Soit une date exprimée en jour, mois et année.

a) définir un type **Date** et écrire les fonctions suivantes :

- ***Permute*** : qui permute deux dates.
- ***Compare*** : qui compare deux dates et retourne 1, 0 ou -1 si D1<D2 ou D1= D2 ou D1>D2 respectivement.

b) Soit une Pile de dates, écrire une fonction ***void Trie(Pile *P)*** qui trie la pile P par ordre croissant (la plus petite date est au sommet de la pile).

Exercice 2 (13 pts) : Le problème suivant concerne les intervalles.

Partie I/ : L'intervalle

Un intervalle d'entiers est défini par sa borne minimale et sa borne maximale, *par exemple* [-3..7].

1. Définir le type ***interval*** pour représenter un intervalle d'entiers.
2. Ecrire la fonction ***interval creerInterval(int min, int max)*** qui retourne l'intervalle [min..max].
3. Ecrire la fonction ***int intervalVide(interval I)*** qui retourne 1 si I =$\emptyset$, (min > max) et 0 sinon.
4. Ecrire les fonctions ***int cardinalite(interval I)*** qui retourne le nombre d'éléments de I.
5. Ecrire la fonction ***int appartient (int x, interval I)*** qui retourne 1 si x $\in$ I et 0 sinon.
6. Ecrire la fonction ***int inclus(interval I1, interval I2)*** qui retourne 1 si $I_1 \subseteq I_2$ et 0 sinon.

   Par exemple : [-2..5] $\subseteq$ [-3..8].

7. Ecrire la fonction ***int memeInterval(interval I1, interval I2)*** qui vérifie si l'intervalle I_1 est égal ou non à l'intervalle I_2.
8. Ecrire la fonction ***interval intersection (interval I1, interval I2)*** qui retourne l'intervalle $I_1 \cap I_2$.

   Par exemple : si I_1 = [-3..7] et I_2 = [-2..10] alors intersection (I1, I2) retourne I = [-2..7].
9. Ecrire la fonction ***int chevauche (interval I1, interval I2)*** qui retourne 1 si $I_1 \cap I_2 \neq \emptyset$ et 0 s'ils sont disjoints.
10. Ecrire la fonction ***interval union (interval I1, interval I2)*** qui retourne l'intervalle $I_1 \cup I_2$. On suppose que I_1 et I_2 chevauchent ($I_1 \cap I_2 \neq \emptyset$).

Par exemple : si I_1 = [-3..7] et I_2 = [-2..10] alors union (I1, I2) retourne I = [-3..10].

Partie II/ : L'ensemble d'intervalles

1. Définir le type ***ensemble*** qui représente un ensemble d'intervalles par une structure de liste. Un élément de la liste est un intervalle.
2. Ecrire la fonction ***void inserer (ensemble *L, interval I)*** qui insère un intervalle I donné dans un ensemble ***L*** trié par ordre croissant de la borne ***min***. Tous les intervalles de ***L*** sont disjoints.
3. Ecrire une fonction ***void supprimer (ensemble *L, interval I)*** qui recherche un intervalle ***I*** donné et le supprime de ***L***.
4. Ecrire une fonction ***ensemble sousEnsemble(ensemble L, interval I)*** qui retourne un sous ensemble ***SL*** qui va contenir les intervalles de ***L*** qui chevauchent avec ***I***.
5. Ecrire une fonction ***interval fusion (ensemble SL, interval I)*** qui, étant donné un ensemble ***SL*** dont tous les éléments chavauche avec ***I***, retourne l'intervalle contenant ***I*** et tous les éléments de ***SL*** (union de tous ces intervalles).

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)

{

}