TD13 + TD14 - soutien - Jeu de UNO

Le but du jeu est de se débarrasser de ses cartes (initialement sept par joueur) en jouant l'un après l'autre. Les 108 cartes sont caractérisées par une valeur (chiffre de 0 à 9 plus deux valeurs particulières : « sauter le tour », qui fait passer le tour du joueur suivant et « changer de sens », qui change le sens du jeu) et une couleur (rouge, vert, bleu ou jaune). A chaque tour, le joueur peut se séparer d'une carte uniquement si elle possède la même couleur ou la même valeur que la précédente. Un joueur ne pouvant jouer doit piocher une carte : si celle-ci est « jouable », elle peut alors être jouée immédiatement. Les joueurs qui terminent sont retirés du jeu au fur et à mesure et leur nom est affiché. La partie est terminée quand il ne reste plus qu'un seul joueur. On utilisera les structures de données suivantes :

```
Type carte = Enregistrement
               val, coul : chaîne
               cartesuiv : ^carte
             FinEnregistrement
     joueur = Enregistrement
               nom : chaîne        {nom du joueur}
               lcarte : ^carte     {liste des cartes du joueur}
               jsuiv, jprec : ^joueur
             FinEnregistrement
     liste-joueur = ^joueur {liste doublement chaînée circulaire}
     sens-suiv = booléen {=vrai si on suit jsuiv, =faux si on suit jprec}
     defausse = pile de carte {pile de cartes jetées}
     pioche = pile de carte {pile de cartes à piocher}
```

1. Ecrivez une procédure qui ajoute une carte dans le jeu du joueur pointé par pj

```
Procédure ajouteCarte (E c : carte, pj : ^joueur)
Var q : ^carte
Début
  q←allouer(carte)
  q^.val←c.val
  q^.coul←c.coul
  q^.cartesuiv←pj^.lcarte
  pj^.lcarte←q
Fin
```

2. Ecrivez une procédure qui supprime la i-ème carte du joueur pointé par pj et l'ajoute sur la défausse.

```
Procédure supprimeCarte (E i : entier, pj : ^joueur, S d : defausse)
Var p,q : ^carte
    j : entier
Début
  q←pj^.lcarte
  Si i=1
    Alors pj^.lcarte←q^.cartesuiv
          empiler(d,q^)
          récupérer(q)
    Sinon Pour j←1 à i-2 inc +1 faire
            q←q^.cartesuiv
          FinPour
          p←q^.cartesuiv
          q^.cartesuiv←p^.cartesuiv
          empiler(d,p^)
          récupérer(p)
  FinSi
Fin
```

3. Ecrivez une procédure qui supprime le joueur pointé par pj.

```
Procédure supprimeJoueur (E pj : ^joueur, S lj : liste-joueur)
Début
  pj^.jprec^.jsuiv←pj^.jsuiv
  pj^.jsuiv^.jprec←pj^.jprec
  Si lj=pj
     Alors lj←pj^.jsuiv
  FinSi
  récupérer(pj)
Fin
```

4. Ecrivez une procédure qui distribue les 7 cartes de départ (contenues dans le jeu) pour chacun des n joueurs (dont on demandera les noms à l'utilisateur) et en créant la liste lj. A chaque fois qu'une carte est distribuée, la procédure `AjouteCarte` est appelée.

```
Procédure distribue (E n : entier, E/S p : pioche, S lj : liste-joueur)
Var q :^joueur
Début
  Pour i←1 à n inc +1 Faire
    écrire ('Entrez le nom du joueur : ')
    lire(nomj)
    q←allouer(joueur)
    q^.nom←nomj
    q^.lcarte←nil
    Si lq=nil
       Alors q^.jsuiv←q
             q^.jprec←q
       Sinon q^.jsuiv←lj
             q^.jprec←lj^.jprec
             lj^.jprec←q
             lj^.jprec^.jsuiv←q
    FinSi
    lj←q
  FinPour
  q←lj
  i←1
  Tantque i≤7 Faire
    ajouteCarte (sommet(p),q)
    dépiler(p)
    q←q^.jsuiv
    Si q=lj
       Alors i←i+1
    FinSi
  FinTantQue
Fin
```

5. Ecrire une fonction choisir carte qui renvoie la position de la première carte possible à jouer dans la liste de cartes du joueur pointé par pj en fonction de la dernière carte jetée sur la défausse ; 0 si aucune carte n'est possible.

```
Fonction ChoisitCarte (pj : ^joueur, d : defausse) : entier
Var res,i : entier
    c : carte
Début
  q←pj^.lcarte
  res←0
  i←1
  c←sommet(d)
  TantQue q≠nil et res=0 Faire
     Si q^.val=c.val ou q^.coul=c.coul
        Alors res←i
```

```
     FinSi
     i←i+1
     q←q^.cartesuiv
  FintantQue
  Retourner(res)
Fin
```

6. Ecrire le programme en pseudo-langage qui permet de jouer une partie complète.

```
Programme Uno
Var lj,q : liste-joueur
    n : entier
    p : pioche
    d : defausse
    sens-suiv, déjà-vu : booléen

Début
  écrire('Combien de joueurs ?')
  lire(n)
  initialise(p)
  distribue (n,p,lj)
  q←lj
  empiler(d,sommet(p))
  dépiler(p)
  sens-suiv←vrai
  déjà-vu←faux
  TantQue lj^.jsuiv≠lj Faire
    Si sommet(d).val= 'changer de sens' et ¬déjà-vu
       Alors Si sens-suiv
                alors q←q^.jsuisv^.jsuiv
                sinon q←q^.jprec^.jprec
             FinSi
             sens-suiv←¬sens-suiv
             déjà-vu←vrai
       Sinon Si sommet(d).val= 'sauter le tour' et ¬déjà-vu
                Alors déjà-vu←vrai
                Sinon i←choisitCarte(q,d)
                      Si i=0
                         Alors ajouteCarte(sommet(p),q)
                               dépiler(p)
                               i←choisitCarte(q,d)
                      FinSi
                      Si i≠0
                          Alors supprimeCarte(i,q,d)
                                Si q^.lcarte=nil
                                   Alors écrire(q^.nom, ' a terminé)
                                         supprimeJoueur(q,lj)
                                FinSi
                                Si déjà-vu
                                    alors déjà-vu←faux
                                FinSi
                      FinSi
             FinSi
    FinSi
    Si sens-suiv
       alors q←q^.jsuisv
       sinon q←q^.jprec
    FinSi
  FinTantQue
Fin
```