DEVOIR SEMESTRE1

Matière : *INFORMATIQUE*
Classes : *2ème Année* ***Durée :*** *1 h*
Préparation : MP, PC et PT

Exercice 1 *(8 pts :1+1+1+5)*

Pour vérifier si une chaîne de caractères est bien parenthésée, c'est-à-dire qu'elle possède autant de parenthèses ouvrantes que de parenthèses fermantes et ceci dans le bon ordre, on utilise une pile.

N.B. : On désigne par le terme « parenthèse », une parenthèse, un crochet ou une accolade.

L'analyse de la chaîne de caractères se fera caractère après caractère. On utilisera une pile contenant les parenthèses ouvrantes.

Toute parenthèse fermante rencontrée doit être associée dans la pile à la parenthèse (qui doit donc exister !) ouvrante de même nature située au sommet de la pile.

Pour chaque caractère, de la chaîne passée en argument, il y a trois possibilités :

- Le caractère n'est pas une parenthèse : dans ce cas, on ne fait rien et on traite l'éventuel caractère suivant.
- Le caractère est une parenthèse ouvrante : dans ce cas, la parenthèse ouvrante est empilée dans la pile.
- Le caractère est une parenthèse fermante, on doit vérifier :
  - que la pile des parenthèses ouvrantes est non vide.
  - que le sommet de la pile est une parenthèse ouvrante de même nature que la parenthèse fermante rencontrée.

  Si ces deux conditions sont vérifiées, on dépile.
- A la fin du parcours, il faut que la pile soit vide pour dire que l'expression est bien parenthésée.

N.B. : Pour le traitement des piles on demande d'utiliser directement les fonctions : empiler(p,e) ; depiler(p) ; est_vide(p) ; sommet(p), pour effectuer respectivement les opérations d'empiler un élément e dans la pile p, de dépiler une pile p en retournant l'élément dépilé, de vérifier si une pile p est vide et de retourner le sommet d'une pile p.

1. Ecrire une fonction *ParOuverte(c)* qui retourne True si « c» est une parenthèse ouvrante et False sinon.
2. Ecrire une fonction *ParFermee(c)* qui retourne True si « c » est une parenthèse fermante et False sinon.

3. Ecrire une fonction *ParAssocie(P)* qui retourne pour une parenthèse fermante « P » la parenthèse ouvrante qui lui est associée.

4. Ecrire une fonction *VerifPar(ch)* qui affiche un message signalant si une chaîne de caractères est bien parenthésée.

Exercice 2 *(12 pts : 2*6)*

On relève dans certains jours de l'année les degrés de température. Une observation sera une liste de 3 valeurs [jour , mois , degré]. Par exemple, une température de 20°C enregistrée le 13 février correspond à la liste [13, 2, 20]. Pour simplifier on supposera que toutes les observations sont enregistrées pendant des années non bissextiles (le nombre de jours du mois de février d'une année non bissextile est égal à 28).

On suppose que les listes *Nb_jours* et *Mois* suivantes sont préalablement définies :

Nb_jours=[31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

Mois=['Janvier','Février','Mars','Avril','Mai','Juin','Juillet','Août','Septembre','Octobre', 'Novembre','Décembre']

1. Ecrire une fonction *Mois_jour()* qui, à partir des deux listes Nb_jour et Mois, retourne un dictionnaire qui associe à chaque mois de la liste *Mois* le nombre de jours correspondant.

   Valeur retournée de la fonction Mois_jour():{"Janvier" :31,"Fevrier" :28, ...,"Décembre":31}

2. Ecrire une fonction *ordre_Mois()* retournant un dictionnaire qui associe à chaque mois de la liste *Mois*, son numéro dans l'année.

   Valeur retournée de la fonction ordre_Mois(): {"Janvier" :1,"Fevrier" :2, ...,"Décembre":12}

3. Ecrire une fonction *valide(obs)* qui retourne True si l'observation *obs* correspond à un jour et un mois valide et False sinon. (Mois valide : compris entre 1 et 12. Jours valide : compris entre 1 et le nombre de jours max du mois)

4. Ecrire la fonction *Num_jour* qui permet de retourner le numéro du jour dans l'année d'une observation *obs* donnée et supposée valide.

   Exemple : l'observation [12, 2, 10] enregistrée le 12 février correspond au 43^{ième} jour de l'année.

5. Ecrire la fonction *Avant* qui, à partir de deux observations *obs1* et *obs2* supposées valides, retourne True si l'observation *obs1* a eu lieu avant l'observation *obs2* et retourne False sinon. (utiliser impérativement la fonction *Num_jour*)

6. Ecrire la fonction *Tri_Obs* qui ordonne une liste d'observations *L* par ordre chronologique des observations. (utiliser impérativement la fonction *Num_jour et le tri par sélection*)