Licence 2, ASDA

Contrôle Semestriel (2018) Durée : 1 heures.

Enoncé de TP

Soit l'expression suivante :

$$a+b\ ,\ (a+b)/d\ ,\ ((c+d)\ +(d-e))+5,\ -(a+b)+(5+b)c,\ -(((a+b)+(c-d))/5)+a5$$

Soit à analyser des expressions mathématiques qui utilisent les symboles (, { et pour l'ouverture des sous expressions et les symboles), } et pour leur fermeture respective. Chaque symbole de fermeture doit être associé à son symbole d'ouverture. Utiliser le modèle de pile pour écrire le programme qui effectue une telle vérification.

Evaluation

La note du contrôle (NC) s'évalue selon le tableau ci-dessous :

Note	Points
Structure	**11**
• Déclaration de la structure de données (classes, listes, ..)	**02**
• Déclaration des méthodes	**03**
• Partie code des méthodes	**03**
• La logique de résolution du problème	**1.5**
• Execution du programme	**1.5**
Lisibilité	**02**
• Lisibilité de code source	**01**
• Commentaires , nom significatif des variables	**01**
Astuces	**03**
• Utilisation des techniques pour compacter le code	**1,5**
• Menu Utilisateur	**1,5**

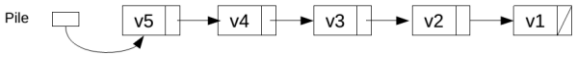
Evaluation finale de l'étudiant

- La note finale (NE) de l'étudiant est évaluée selon la formule suivante :

$$NE = NC + Bonus - Malus$$

- Un bonus est un ou plusieurs points à ajouter à la note totale dans le cas où :
  - Travaux demandés, Assiduité (Efficacité), Continuité dans le travail (+4)
- Un malus est un ou plusieurs points à déduire dans le cas où :
  - Un programme non cohérent (-2)
  - Erreurs ou omissions seront prisent en compte (-1 , -3)
  - Le code source non lisible et de mauvaise qualité (-2)
  - L'étudiant a des absences non justifiées (-0,5, -4)
- **NB** *: Attention au plagiat de code : si vous empruntez ou si vous vous inspirez d'un code trouvé sur internet, vous devez impérativement indiquer la provenance et justifier la raison.*

Annexe : Implémentation d'une Pile en dynamique




```
struct maillon {
int val;
struct maillon *adr;
};
typedef struct maillon  *Pile;

void CreerPile ( Pile  *p)
 { *p=0;}

int PileVide (Pile p)
{ return (P==0);}

int PilePleine(Pile  p)
{ return 0;}

int Empiler(int x,Pile *p) {
   struct maillon *q;
   if (PilePleine(*p))
    return 0;
     q=malloc(sizeof(*q));
     q->val= x;
     q->adr =*p;
    *p= q;
     return 1;
}
int Depiler(int  *x, Pile  *p ) {
            struct maillon *q;
            if (PileVide(*p))return0;
            *x= (*p)->val;q = *p;*p= (*p)->adr;
     free (q);
return 1;
}

int main ()
{
……..

   return 0;
}
```