Examen de Rattrapage
Algorithmique

Exercice 1(5pt)

Soit la multiplication de deux entiers positifs a et b selon le principe récursif suivant :

a * b = a * (b-1) + a **si b est impair**

a * b = (a+a) * (b/2) **si b est pair**

Le traitement s'arrête **si b = 0**

Exemple : 36 * 7 = 36 *6 +36 = 72 *3 +36

= 72*2 +108= 144 *1 +108

= 144*0+252 = 252

1) Ecrire une fonction récursive qui permet de multiplier deux nombres par la méthode décrite ci-dessus.

2) Ecrire une fonction itérative qui permet de réaliser le même traitement.

Exercice 2: (5pts)

Soit T un vecteur de nombres réels de taille n (n<=1000) et soit x un nombre réel donné.

1. Ecrire une action paramétrée Somme_2 qui vérifie s'il existe deux éléments de T dont la somme est égale à x. Donner sa complexité en justifiant votre réponse.

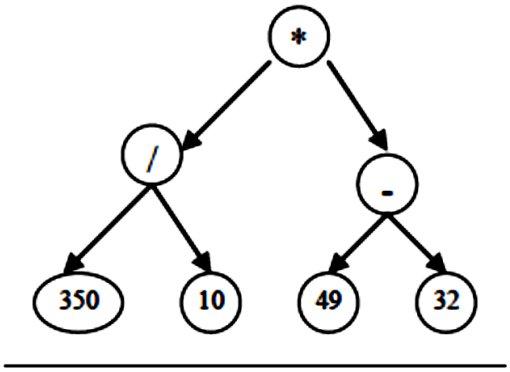
On suppose maintenant que le vecteur T est trié dans l'ordre croissant.

2. Ecrire une action paramétrée Somme_2bis qui résout la question n°1 avec une complexité O(n). Justifiez votre réponse.

Exercice 3 (10 pts)

Soit une expression arithmétique représentée par un arbre binaire où les nœuds sont des chaines de caractères (de longueur <= 5) correspondant à un *opérateur* ou un *opérande*. Pour simplifier, on ne considère que les opérandes entiers et les opérateurs +, -, *, /

Exemple



L'arbre correspond à l'expression (350/10) *(49-32) qui vaut 595

1. Ecrire une fonction **Opérateur** qui accepte en entrée un caractère c et vérifie si c'est un opérateur (+, -, *, /) ou non.

2. Ecrire une fonction **Opération** qui accepte en entrée deux opérandes *op1* et *op2* de type *entier* et un opérateur *op* de type *caractère*, calcule et renvoie le résultat de l'opération (op1 **op** op2).

3. A partir de l'arbre contenant l'expression arithmétique, on voudrait récupérer la forme ***postfixée*** de l'expression dans une pile de chaines de caractères.

- Donner la définition du type *arbre* précédemment décrit.
- Donner la définition du type pile de chaines de caractères
- Ecrire les primitives **Empiler** et **Dépiler**.
- Ecrire une fonction **Postfixe** qui remplit la pile à partir de l'arbre, en effectuant un parcours postfixé.

4. Ecrire une fonction **Eval** qui évalue l'expression arithmétique à partir de la pile construite.

On utilisera la procédure ***Val(s : chaine de caractères ; var nbr :reel +0)*** pour transformer une chaine de caractères **s** (numériques) en l'entier correspondant **nbr** .

5. On suppose que l'arbre de l'expression arithmétique est déjà construit.

Ecrire l'algorithme qui construit la pile contenant la forme postfixée, évalue l'expression et affiche le résultat de l'évaluation.

Corrigé

Exo1

Fonction produit(X, Y: entier):entier

Si Y = 0

alors produit← 0

Sinon

si Y mod 2 =0 alors produit← produit (X+X, Y div 2)

Sinon produit ← produit(X, Y-1) +Y

Fin si

Fin si

Fin produit ;

Fonction produit2((X, Y: entier):entier

Debut

P :=0 ;

Tant que Y<>0

si X mod 2 =0 alors X :=X+X ;Y :=Ydiv2

Sinon Y :=Y-1 ;P :P+X ;

Fin si ;

Produit2 :=P ;

Fin produit2 ;

Exo 3

1)

Fonction Operateur(char :caractère):bouleen;

```
Debut
 si ((ch== '+')  ou (ch == '-') ou (ch== '*') ou (ch=='/'))
        alors Operateur :=vraie
     sino
Operateur :=faux
Fin ;
```

2)

Fonction operation(x,y :entier ;op :caractère) :reel ;

```
Debut
   case  op of
   '+' :  operation:= x + y;
   '-' :  operation:= x - y;
'*' :  operation:= x * y;
'/' :  operation:= x / y;
   Fincase;
```

Fin operation;

3)

Type arbre	**Type pile**
Arbre=^elemnt1 Element1= record Val1: caractère; Fg,Fd: arbre ; end ;	**Pile=**^element2 Element2= **record** **val2:** caractère; **suiv**: pile end ;

Procédure empiler	**Procédure dépiler**
Procedure Empiler (var Tête : Pile ; ch:caractères) var P : Pile DEBUT Allouer(P) ; P^.val2 := ch ; P^.Suivant : = Tête ; Tête := P Fin ;	Procedure Empiler (var Tête : Pile, ch:caractères) var P : Pile DEBUT ch := P^.val2; P = Tête ; Tête := P^.Suivant ; Dispose (p) ; Fin ;

Procédure postfixe(a :arbre ;var p:pile) ;

```
Debut   Si a<>nil
        Alors postfixe(a^.Fg) ;
              postfixe(a^.Fd) ;
             empiler(p,a^,val1) ;
        fsi
fin ;
```

```
fonction ÉvaluerPF(expr):valeur ;
Var
  I,nb1,nb2: entier ;x,y :caractère ;
Z :reel ;
  p: pile
DEBUT
  i ← 1
 initpile( p)
  TQ (i ≤lenght(exp)) FAIRE
    SI operateur(expr[i]) ALORS
      Dépiler(p,x) ;val(x,nb1) ;
      Dépiler(p,y) ;val(y,nb2) ;
      z ← Operation(nb1,nb2,expr[i])
      Empiler(p,z)
    SINON
      Empiler(p,expr[i])
    FSI
    i ← i + 1
  FTQ
 Evaluer :=Depiler(p) ;
FIN ;
Algorithme
Debut
Postfix(a,p) ;
PF :=evaluerPF(exp) ;
Ecrire(pf) ;
Fin .
```