

* Q1

====

voir fichier joint

* Q2

====

Une erreur lexicale consiste à utiliser un lexème incorrectement écrit ou inexistant,
par exemple :

(sete V1 42)

* Q3

====

Le programme suivant comporte une erreur syntaxique (oubli de la parenthèse ouvrante) :

set V1 42

* Q4 et Q5

=====

```
void Rec_Pgm(Ast *A) {  
    demarrer() ;  
    Rec_Par_Inst(A) ;  
    arreter() ;  
}
```

```
void Rec_Par_Inst(Ast *A) {  
    if (LC().nature == PARO) {  
        Avancer() ;  
        Rec_Inst(A) ;  
        if (LC().nature == PARF)  
            Avancer() ;  
        else  
            Erreur() ;  
    } else {  
        Erreur() ;  
    } ;  
}
```

```
void Rec_Inst(Ast *A) {  
    Ast Ag, Ad ;  
    switch (LC().nature) {  
        case SET:  
            Avancer() ;  
            if (LC().nature == VAR) {  
                Ag = creer_var(LC().chaine) ;  
                Avancer() ;  
            } else  
                Erreur() ;  
            Rec_Exp(&Ad) ;  
            *A = creer_set(Ag, Ad) ;  
            break ;  
        case SEQ:  
            Avancer() ;  
            Rec_Par_Inst(&Ag) ;  
            Rec_Par_Inst(&Ad) ;  
            *A = creer_seq(Ag, Ad) ;  
            break ;  
    }
```

```

        default:
            Erreur() ;
    }
}

void Rec_Exp(Ast *A) {
    Ast Ag, Ad ;
    switch (LC().nature) {
        case ENTIER:
            *A = creer_entier(LC().val) ;
            Avancer() ;
            break ;
        case ADD:
            Avancer() ;
            if (LC().nature == VAR) {
                Ag = creer_var(LC().chaine) ;
                Avancer() ;
            } else
                Erreur() ;
            if (LC().nature == VAR) {
                Ad = creer_var(LC().chaine) ;
                Avancer() ;
            } else
                Erreur() ;
            *A = creer_add (Ag, Ad) ;
            break ;
        case SUB:
            // idem ADD en remplaçant creer_add par creer_sub ...
        default:
            Erreur() ;
    }
}

* Q6
=====

void executer (A) {
    intTS() ;
    exec (A) ;
    afficheTS() ;
}

void exec (A) {
    int val ;
    switch (A.nature) {
        case N_SEQ:
            exec (A.fg) ;
            exec (A.fd) ;
            break ;
        case N_SEQ:
            val = eval(A.fd)
            set (A.fg.chaine, v) ;
            break ;
        default:
            Erreur() ;
    }
}

int evaluer (A) {
    int val ;
    switch (A.nature) {
        case N_ENTIER:

```

```
        return A.val ;
case N_ADD:
    vg = val(A.fg.chaine) ;
    vd = val(A.fd.chaine) ;
    return vg+vd ;
case N_SUB:
    vg = val(A.fg.chaine) ;
    vd = val(A.fd.chaine) ;
    return vg-vd ;
default:
    Erreur() ;
    }
}
```