1.
```
int motif(char * pattern, char * texte)
      // Les positions sont comptées comme en Pascal : de 1 à la
      // longueur, et non pas à partir de 0
{
int i, j ;
int lp, lt ;               // longueurs des chaînes

  // On vérifie qu'aucune chaîne n'est vide
  if (pattern[0]=='\0' || texte[0]=='\0') return -1 ;

  // Le texte doit être plus long que le pattern
  lp = strlen(pattern) ; lt = strlen(texte) ;
  if (lp > lt) return -2 ;

  // On est dans le cas normal
  i=0 ;
  do {
    for (j=0 ; j < lp && pattern[j]==texte[i+j] ; j++)  ;
    i++ ;
  } while (i<=(lt-lp) && j < lp) ;

  if (j < lp) return 0 ;
  else        return i ;
}
```

2. *Cf.* cours.

3.
procedure ***Parcours***$(\mathcal{A})$;
  var X : sommet ;
  début
    X := `racine`$(\mathcal{A})$;
    répéter
      si `existe-fils`(X) **alors**
        Empiler(X) ;
        X := `premier-fils`(X) ;
      sinon {
        `(* le sommet courant n'a pas de fils : c'est une feuille *)`
        afficher(X)
        tant que { X $\neq$ `racine`$(\mathcal{A})$ **et** **non** `existe-frère`(X) } **faire**
          X := `dépiler()` ;
        si `existe-frère`(X) **alors**
          X := `frère`(X) ;
      }
    jusqu'à X := `racine`$(\mathcal{A})$;
  fin ;

4. *Cf.* cours.