1. Il est plus facile de définir d'abord une fonction de comparaison, qui correspond à l'opération de « calque » vue en cours.

```
function compare (i:integer,chaine:string) : boolean
(* i : indice dans le tableau T, chaine : chaîne à calquer sur T[i] *)
var j : integer ;
begin
  j := 1 ;
  while T[i+j-1] = chaine[j] and j <= length(chaine) do
    j := j + 1  ;
  compare := ( T[i+j-1] = chaine[j] ) ;
end ;

begin
  writeln('Balise recherchée ? '); readln(balise) ;
  motif1 := '<' + balise + '>' ;
  motif2 := '</' + balise + '>' ;

  for i:=1 to n do begin
    if compare(i, motif1) then in := TRUE ;
    if in then write(T[i]) ;
    if in and compare(i, motif2) then in := FALSE
  end ;
end ;
```

2. (a)
```
function avant(x, y : string[20]) : boolean ;
var i,j : integer ;
begin
  i := length(x) ; j := length(y) ;
  while x[i] = y[j] and i >= 1 and j >= 1 do
    begin i:=i -1 ; j:=j-1 end;
  if x[i] < y[j] then avant := TRUE
  else                avant := FALSE ;
end ;
```

   (b) Exemple : tri par sélection-échange. Il vaut mieux éviter un tri qui déplace beaucoup les éléments.

```
for i:=1 to N do begin
  min := i ;
  for j := i to N do
    if avant(Tm[j], Tm[min]) then min := j ;
  echange(Tm[i], Tm[min]) ;
end ;
```

3. Bien sûr, pour faire une recherche dichotomique, on suppose le tableau trié.

```
function rech_dicho(val : integer, g,d : integer) : boolean ;
var m : integer ;
begin
  if g < d then begin
    m := g + d div 2 ;
    if tab[m] < val then      rech_dico := rech_dico(val, m+1, d)
    else if tab[m] = val then rech_dico := TRUE
         else                 rech_dico := rech_dico(val, g, m) ;
  end
  else                        rech_dico := ( tab[g] = val )
end
```