ÉCOLE POLYTECHNIQUE

CONCOURS D'ADMISSION 2010 FILIÈRES PSI et PT

ÉPREUVE D'INFORMATIQUE

Ce corrigé a été rédigé sous Maple V.5.1.

Partie I

Question 1

```
> restart:
```

Il semble que sous Maple, la structure la plus commode pour coller à l'énoncé est la structure de liste.

```
> tab:=[4,8,15,2,2,1];
```

$$tab := [4, 8, 15, 2, 2, 1]$$

```
> whattype(tab);
```

$$list$$

```
> tab[3..5];
```

$$[15, 2, 2]$$

On construit les fonctions primitives décrites dans le sujet (elles ne sont pas exigibles du candidat).

```
> allouer:=proc(n)
  local k;
  [seq(0,k=1..n)];
  end;
```

$$allouer := \mathbf{proc}(n)\ \mathbf{local}\ k;\ [\mathrm{seq}(0, k = 1 \mathinner{..} n)]\ \mathbf{end}$$

```
> taille:=t->nops(t);
```

$$taille := nops$$

On prend l'exemple de l'énoncé 'quelbonbonbon' avec un stockage numérique par lettre.

```
> tab:=[17,20,5,12,2,15,14,2,15,14,2,15,14];
  bonbon:=[2,15,14,2,15,14];
```

$$tab := [17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14]$$

$$bonbon := [2, 15, 14, 2, 15, 14]$$

```
> taille(tab);
```

$$13$$

```
> TeteDeSuffixe:=proc(mot,tab,k)
  local res, i;
  res:=true;
  i:=k+taille(mot)-1;
  if i>taille(tab) then res:=false; fi;
  while res and i>=k do
  res:=res and tab[i]=mot[i-k+1];
  i:=i-1;
  od;
  res;
  end;
```

$$TeteDeSuffixe := \mathbf{proc}(mot, tab, k)$$
$$\mathbf{local}\ res, i;$$
$$res := true;$$
$$i := k + \mathrm{taille}(mot) - 1;$$
$$\mathbf{if}\ \mathrm{taille}(tab) < i\ \mathbf{then}\ res := false\ \mathbf{fi};$$
$$\mathbf{while}\ res\ \mathbf{and}\ k \le i\ \mathbf{do}\ res := res\ \mathbf{and}\ tab[i] = mot[i - k + 1];\ i := i - 1\ \mathbf{od};$$
$$res$$
$$\mathbf{end}$$

```
> TeteDeSuffixe([12,2,15],tab,4);
  TeteDeSuffixe([1,0],tab,7);
```

$$true$$

$$false$$

Question 2

```
> rechercherMot:=proc(mot,tab)
  local res,k;
  res:=false;
  k:=1;
  while not(res) and k<=taille(tab)-taille(mot)+1 do
  res:=res or TeteDeSuffixe(mot,tab,k);
  k:=k+1;
  od;
  res;
  end;
```

$$rechercherMot := \mathbf{proc}(mot, tab)$$
$$\mathbf{local}\ res, k;$$
$$res := false;$$
$$k := 1;$$
$$\mathbf{while\ not}\ res\ \mathbf{and}\ k \le \mathrm{taille}(tab) - \mathrm{taille}(mot) + 1\ \mathbf{do}$$
$$res := res\ \mathbf{or}\ \mathrm{TeteDeSuffixe}(mot, tab, k);\ k := k + 1$$
$$\mathbf{od};$$
$$res$$
$$\mathbf{end}$$

```
> tab;
```

[17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14]

```
> rechercherMot([5,8,3],tab);
  rechercherMot([14,2,15,14],tab);
  rechercherMot(tab,tab);
  rechercherMot(bonbon,tab);
```

false

true

true

true

Question 3

```
> compterOccurences:=proc(mot,tab)
  local k,oc;
  k:=1;oc:=0;
  while rechercherMot(mot,tab[k..taille(tab)]) do
  if TeteDeSuffixe(mot,tab,k)
  then oc:=oc+1; fi;
  k:=k+1;
  od;
  oc;
  end;
```

compterOccurences := **proc**(*mot*, *tab*)
local *k*, *oc*;
k := 1;
oc := 0;
while rechercherMot(*mot*, *tab*[*k* .. taille(*tab*)]) **do**
if TeteDeSuffixe(*mot*, *tab*, *k*) **then** *oc* := *oc* + 1 **fi**; *k* := *k* + 1
od;
oc
end

```
> tab;
```

[17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14]

```
> compterOccurences([15],tab);
  compterOccurences([17,20],tab);
  compterOccurences([17,20,0],tab);
  compterOccurences(bonbon,tab);
```

3

1

0

2

Question 4

```
> frequenceLettre:=proc(tab)
  local k,res;
  res:=[];
  for k from 1 to 26 do
  res:=[op(res),compterOccurences([k],tab)];
  od;
  res;
  end;
```

frequenceLettre := **proc**(*tab*)
local *k*, *res*;
res := [ ]; **for** *k* **to** 26 **do** *res* := [op(*res*), compterOccurences([*k*], *tab*)] **od**; *res*
end

```
> frequenceLettre(tab);
```

[0, 3, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 3, 3, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0]

Question 5

```
> afficherMot:=proc(tab,i,k)
  print(tab[i..i+k-1]);
  end;
```

afficherMot := **proc**(*tab*, *i*, *k*) print(*tab*[*i* .. *i* + *k* − 1]) **end**

```
> afficherMot(tab,1,2);
```

[17, 20]

```
> afficherFrequenceBigramme:=proc(tab)
  local k,mot,tabc;
  for k from 1 to taille(tab)-1 do
  mot:=tab[k..k+1];
  tabc:=tab[1..k+1];
  if compterOccurences(mot,tabc)=1 then
  afficherMot(tab,k,2);
  print(compterOccurences(mot,tab));
  fi;
  od;
  end;
```

afficherFrequenceBigramme := **proc**(*tab*)
local *k*, *mot*, *tabc*;
for *k* **to** taille(*tab*) − 1 **do**
mot := *tab*[*k* .. *k* + 1];
tabc := *tab*[1 .. *k* + 1];
if compterOccurences(*mot*, *tabc*) = 1 **then**
afficherMot(*tab*, *k*, 2); print(compterOccurences(*mot*, *tab*))
fi

```
    od
end
> tab;
                    [17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14]
> affichercFrequenceBigramme(tab);
                                  [17, 20]
                                     1
                                  [20, 5]
                                     1
                                  [5, 12]
                                     1
                                  [12, 2]
                                     1
                                  [2, 15]
                                     3
                                  [15, 14]
                                     3
                                  [14, 2]
                                     2
```

Partie II

Question 6

```
> comparerSuffixes:=proc(tab,k1,k2)
  local s1,s2,n,n1,n2,k,res,nmin;
  n:=taille(tab);
  s1:=tab[k1..n];
  s2:=tab[k2..n];
  n1:=taille(s1);
  n2:=taille(s2);
  nmin:=min(n1,n2);
  k:=1;
  while k<=nmin and s1[k]=s2[k] do
  k:=k+1;
  od;
  if k>max(n1,n2) then res:=0 else
    if k>nmin then res:=piecewise(n1<n2,-1,1);
    else res:=piecewise(s1[k]<s2[k],-1,1);fi;
  fi;
  end;
```

comparerSuffixes := **proc**(*tab*, *k1*, *k2*)
local *s1*, *s2*, *n*, *n1*, *n2*, *k*, *res*, *nmin*;
n := taille(*tab*);
s1 := *tab*[*k1* .. *n*];
s2 := *tab*[*k2* .. *n*];
n1 := taille(*s1*);
n2 := taille(*s2*);
nmin := min(*n1*, *n2*);
k := 1;
while *k* ≤ *nmin* **and** *s1*[*k*] = *s2*[*k*] **do** *k* := *k* + 1 **od**;
if max(*n1*, *n2*) < *k* **then** *res* := 0
else
if *nmin* < *k* **then** *res* := piecewise(*n1* < *n2*, -1, 1)
else *res* := piecewise(*s1*[*k*] < *s2*[*k*], -1, 1)
fi
fi
end

```
> tab;
                    [17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14]
> comparerSuffixes(tab,1,2);
  comparerSuffixes(tab,2,taille(tab));
  comparerSuffixes(tab,5,5);
                                     -1
                                      1
                                      0
```

Question 7

```
> calculerSuffixes:=proc(tab)
  local k,n,res,loc,i,bool;
  n:=taille(tab);
  res:=[seq(k,k=1..n)];
  for k from 1 to n-1 do
  i:=k;
  bool:=true;
  while bool and i>0 do
    bool:=comparerSuffixes(tab,res[i],res[i+1])>0;
    if bool then
      loc:=res[i];
      res[i]:=res[i+1];
      res[i+1]:=loc;
    fi;
    i:=i-1;
    od;
  od:
  res;
```

```
    end;
calculerSuffixes := proc(tab)
local k, n, res, loc, i, bool;
    n := taille(tab);
    res := [ seq(k, k = 1 .. n) ];
    for k to n − 1 do
        i := k;
        bool := true;
        while bool and 0 < i do
            bool := 0 < comparerSuffixes(tab, res[ i ], res[ i + 1 ]);
            if bool then loc := res[ i ]; res[ i ] := res[ i + 1 ]; res[ i + 1 ] := loc fi;
            i := i − 1
        od
    od;
    res
end
> tab;
                        [ 17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14 ]
> tabS:=calculerSuffixes(tab);
                    tabS := [ 11, 8, 5, 3, 4, 13, 10, 7, 12, 9, 6, 1, 2 ]
```

On retrouve l'ordre lexicographique donné en exemple.

Partie III

Question 8

```
> comparerMotSuffixe:=proc(mot,tab,k)
  local s1,s2,n,n1,n2,i,res,nmin;
  n:=taille(tab);
  s2:=tab[k..n];
  s1:=mot;
  n1:=taille(s1);
  n2:=taille(s2);
  nmin:=min(n1,n2);
  i:=1;
  while i<=nmin and s1[i]=s2[i] do
  i:=i+1;
  od;
  if i>n1 then res:=0 else
    if i>nmin then res:=piecewise(n1<n2,-1,1);
    else res:=piecewise(s1[i]<s2[i],-1,1);fi;
  fi;
  end;
comparerMotSuffixe := proc(mot, tab
```



```
local s1, s2, n, n1, n2, i, res, nmin;
    n := taille( tab );
    s2 := tab[ k .. n ];
    s1 := mot;
    n1 := taille( s1 );
    n2 := taille( s2 );
    nmin := min( n1, n2 );
    i := 1;
    while i ≤ nmin and s1[ i ] = s2[ i ] do i := i + 1 od;
    if n1 < i then res := 0
    else
        if nmin < i then res := piecewise( n1 < n2, -1, 1 )
        else res := piecewise( s1[ i ] < s2[ i ], -1, 1 )
        fi
    fi
end
> tab;taille(tab);
  mot:=[15,14];
                    [ 17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14 ]
                                        13
                                mot := [ 15, 14 ]
> comparerMotSuffixe(mot,tab,1);
  comparerMotSuffixe(mot,tab,3);
  comparerMotSuffixe(mot,tab,12);
  comparerMotSuffixe([17,20,5],tab,1);
                                        -1
                                         1
                                         0
                                         0
```

Question 9

```
> rechercherMot2:=proc(mot,tab,tabS)
  local k,c,tS,bool;
  tS:=tabS;
  k:=taille(tab);
  if k=1 and mot=tab
  then bool:=true
  else bool:=false; fi;
  while k>1 and not(bool) do
    k:=floor(k/2);
    c:=comparerMotSuffixe(mot,tab,tS[k]);
    if c=0 then bool:=true;
```

```
      else
        if c<0 then tS:=tS[1..k];
        else tS:=tS[k+1..taille(tS)];
        fi;
        k:=taille(tS);
      fi;
      od;
    bool;
    end;
```

rechercherMot2 := **proc**(*mot*, *tab*, *tabS*)
local *k*, *c*, *tS*, *bool*;
  tS := *tabS*;
  k := taille(*tab*);
  if *k* = 1 **and** *mot* = *tab* **then** *bool* := *true* **else** *bool* := *false* **fi**;
  while 1 < *k* **and not** *bool* **do**
    k := floor(1 / 2*k);
    c := comparerMotSuffixe(*mot*, *tab*, *tS*[*k*]);
    if *c* = 0 **then** *bool* := *true*
    else
      if *c* < 0 **then** *tS* := *tS*[1 .. *k*] **else** *tS* := *tS*[*k* + 1 .. taille(*tS*)] **fi**;
      k := taille(*tS*)
    fi
  od;
  bool
end

```
> tab;tabS;
```

[17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14]

[11, 8, 5, 3, 4, 13, 10, 7, 12, 9, 6, 1, 2]

```
> rechercherMot2([15,14,2,15,14],tab,tabS);
  rechercherMot2([17,20,5],tab,tabS);
  rechercherMot2([20,5,3],tab,tabS);
  rechercherMot2([17,20,5,4],tab,tabS);
  rechercherMot2(tab,tab,tabS);
  rechercherMot2([5,12,2,15,14],tab,tabS);
  rechercherMot2(bonbon,tab,tabS);
```

true
true
false
false
true
true

true

Question 10

Dans la procédure rechercherMot, on parcourt le tableau case par case en comparant chaque suffixe démarrant sur la case courant. On a donc un nombre de comparaisons de l'ordre de n (cas le plus défavorable où il faut parcourir tout le tableau).
Dans la procédure rechercherMot2, le nombre de comparaisons est log_2(n) car le tableau sur lequel s'effectue la comparaison voit sa taille divisée par 2 à chaque étape de la dichotomie.
On préfère donc cette deuxième procédure qui possède la meilleure complexité algorithmique.

Question 11

```
> rechercherPremierSuffixe:=proc(mot,tab,tabS)
  local k,c,tS,indS,bool;
  tS:=tabS;
  indS:=[seq(k,k=1..taille(tabS))];
  k:=taille(tab);
  while k>1 do
    k:=floor(k/2);
    c:=comparerMotSuffixe(mot,tab,tS[k]);
    if c<=0
    then
       tS:=tS[1..k];
       indS:=indS[1..k];
    else
       tS:=tS[k+1..taille(tS)];
       indS:=indS[k+1..taille(indS)];
    fi;
    k:=taille(tS);
  od;
  if comparerMotSuffixe(mot,tab,op(tS))=0 then op(indS) else
  0 fi;
  end;
```

rechercherPremierSuffixe := **proc**(*mot*, *tab*, *tabS*)
local *k*, *c*, *tS*, *indS*, *bool*;
  tS := *tabS*;
  indS := [seq(*k*, *k* = 1 .. taille(*tabS*))];
  k := taille(*tab*);
  while 1 < *k* **do**
    k := floor(1 / 2*k);
    c := comparerMotSuffixe(*mot*, *tab*, *tS*[*k*]);
    if *c* ≤ 0 **then** *tS* := *tS*[1 .. *k*]; *indS* := *indS*[1 .. *k*]
    else *tS* := *tS*[*k* + 1 .. taille(*tS*)]; *indS* := *indS*[*k* + 1 .. taille(*indS*)]

```
        fi;
        k := taille(tS)
    od;
    if comparerMotSuffixe(mot, tab, op(tS)) = 0 then op(indS) else 0 fi
end
```

```
> tab;tabS;
```

$$[17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14]$$

$$[11, 8, 5, 3, 4, 13, 10, 7, 12, 9, 6, 1, 2]$$

```
> rechercherPremierSuffixe([5],tab2,tab2S);
```

$$0$$

```
> rechercherPremierSuffixe([2,15,14],tab,tabS);
  rechercherPremierSuffixe([14,2,15,14],tab,tabS);
  rechercherPremierSuffixe([3,15,14],tab,tabS);
  rechercherPremierSuffixe(bonbon,tab,tabS);
```

$$1$$

$$7$$

$$0$$

$$2$$

Question 12

```
> rechercherDernierSuffixe:=proc(mot,tab,tabS)
  local k,c,tS,indS,bool;
  tS:=tabS;
  indS:=[seq(k,k=1..taille(tabS))];
  k:=taille(tab);
  while k>1 do
    k:=floor(k/2);
    c:=comparerMotSuffixe(mot,tab,tS[k+1]);
    if c<0
    then
       tS:=tS[1..k];
       indS:=indS[1..k];
    else
       tS:=tS[k+1..taille(tS)];
       indS:=indS[k+1..taille(indS)];
    fi;
    k:=taille(tS);
  od;
  if comparerMotSuffixe(mot,tab,op(tS))=0 then op(indS) else
  0 fi;
  end;
```

```
rechercherDernierSuffixe := proc(mot, tab, tabS)
local k, c, tS, indS, bool;
    tS := tabS;
```



```
    indS := [seq(k, k = 1 .. taille(tabS))];
    k := taille(tab);
    while 1 < k do
        k := floor(1 / 2*k);
        c := comparerMotSuffixe(mot, tab, tS[k + 1]);
        if c < 0 then tS := tS[1 .. k]; indS := indS[1 .. k]
        else tS := tS[k + 1 .. taille(tS)]; indS := indS[k + 1 .. taille(indS)]
        fi;
        k := taille(tS)
    od;
    if comparerMotSuffixe(mot, tab, op(tS)) = 0 then op(indS) else 0 fi
end
```

```
> tab;tabS;mot;
```

$$[17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14]$$

$$[11, 8, 5, 3, 4, 13, 10, 7, 12, 9, 6, 1, 2]$$

$$[15, 14]$$

```
> rechercherDernierSuffixe(mot,tab,tabS);
  rechercherDernierSuffixe(bonbon,tab,tabS);
```

$$11$$

$$3$$

```
> compterOccurences2:=(mot,tab,tabS)->rechercherDernierSuffi
  xe(mot,tab,tabS)-rechercherPremierSuffixe(mot,tab,tabS)+1;
```

$$compterOccurences2 := (mot, tab, tabS) \rightarrow \text{rechercherDernierSuffixe}(mot, tab, tabS) - \text{rechercherPremierSuffixe}(mot, tab, tabS) + 1$$

```
> compterOccurences2(mot,tab,tabS);
  compterOccurences2(bonbon,tab,tabS);
```

$$3$$

$$2$$

Question 13

```
> afficherFrequenceKgramme:=proc(tab,tabS,k)
  local i,mot,tabc;
  for i from 1 to taille(tab)-k+1 do
  mot:=tab[i..i+k-1];
  tabc:=tab[1..i+k-1];
  if compterOccurences2(mot,tabc,calculerSuffixes(tabc))=1
  then
  afficherMot(tab,i,k);
  print(compterOccurences2(mot,tab,tabS));
  fi;
  od;
  end;
```

```
afficherFrequenceKgramme := proc(tab, tabS, k)
local i, mot, tabc;
    for i to taille(tab) − k + 1 do
        mot := tab[ i .. i + k − 1 ];
        tabc := tab[ 1 .. i + k − 1 ];
        if compterOccurences2(mot, tabc, calculerSuffixes(tabc)) = 1 then
            afficherMot(tab, i, k); print(compterOccurences2(mot, tab, tabS))
        fi
    od
end
```

Pas sûr que cette procédure soit satisfaisante. Elle est clonée sur celle de la question 5 et utilise le tableau des suffixes uniquement pour compter les occurences. En tout cas, elle fonctionne.

```
> tab;tabS;
```

[17, 20, 5, 12, 2, 15, 14, 2, 15, 14, 2, 15, 14]

[11, 8, 5, 3, 4, 13, 10, 7, 12, 9, 6, 1, 2]

```
> afficherFrequenceKgramme(tab,tabS,4);
```

[17, 20, 5, 12]
1
[20, 5, 12, 2]
1
[5, 12, 2, 15]
1
[12, 2, 15, 14]
1
[2, 15, 14, 2]
2
[15, 14, 2, 15]
2
[14, 2, 15, 14]
2

```
> afficherFrequenceKgramme(tab,tabS,3);
```

[17, 20, 5]
1
[20, 5, 12]
1
[5, 12, 2]
1
[12, 2, 15]
1

[2, 15, 14]
3
[15, 14, 2]
2
[14, 2, 15]
2