Corrigé colle d'informatique n°2.

Épreuve facultative d'informatique de l'X - 2002 - MP/PC

Comme tenu de l'implémentation des tableaux en Maple, on introduit la fonction `longueur` suivante valable pour toute les questions :

```
longueur := proc(a :: array)
    return(nops(convert(a,list));
end;
```

Question 1 : Une idée pour calculer l'amplitude est de parcourir une fois l tableau afin de repérer le minimum (variable `m`) et le maximum (variable `M`) du tableau est de retourner l'amplitude donnée par `M-m`. On balaye une seule fois le tableau, notre fonction s'exécute en temps linéaire.

```
amplitude := proc(a :: array)
  local i, n, m, M;
  M:= a[0]; m:= a[0];
  n := longueur(a);
  for i from 1 to n-1 do
     if a[i] < m then m := a[i] fi;
     if a[i] > M then M := a[i] fi;
  od;
  return(M-m)
end;
```

Question 2 : Il suffit de considérer un cours décroissante de valeurs. Le gain sera nul alors que l'amplitude sera strictement positive.

Question 3 :

```
gain := proc(a :: array)
  local g, i, j, n;
  g := 0;
  n := longueur(a); # On récupère la longueur du tableau
  for i from 0 to n-2 do
    for j from i+1 to n-1 do
     if a[j] - a[i] > g then g := a[j] - a[i] fi;
    od;
  od;
  return(g)
end;
```

Question 4 :

```
gain := proc(a :: array)
  local g, i, j, n, i0, j0;
  g := 0;
  i0 := 0; # pour stocker l'instant d'achat
  j0 := 0; # pour stocker l'instant de vente
  n := longueur(a); # On récupère la longueur du tableau
  for i from 0 to n-2 do
    for j from i+1 to n-1 do
     if a[j] - a[i] > g
             then g := a[j] - a[i]; i0 := i; j0 := j fi;
     if a[j] - a[i] = g and ((j-i) < (j0-i0))
            then i0 := i; j0 := j fi; # on actualise i0 et j0
    od;
  od;
  printf("Instant d'achat i = %d et instant de vente j = %d\n",i0,j0);
  return(g)
end;
```

Question 5 :

```
gain1 := proc(a::array)
  local i, i0, n, g, k;
  n := longueur(a); # On récupère la longueur du tableau
  g := 0;  # Le gain est nul au départ
  i0 := 0; # permet de repérer l'instant d'achat
  k := 0;  # permet de repérer l'instant où la cote d'action
           # est la plus basse.
  for i from 1 to n-1 do
     if a[i] - a[i0] > g then g := a[i] - a[i0]; fi;  #
     if a[i] - a[k] > g then g := a[i] - a[k]; i0 := k; fi; #
     if a[i] < a[k] then k := i fi; # On change k si la cote
             # à l'instant i est plus basse qu'à l'instant k
  od;
  return(g);
end;
```

Question 6 :

```
gain1 := proc(a::array)
 local i, i0, j0, n, g, k, delta;
  n := longueur(a); # On récupère la longueur du tableau
  g := 0;  # Le gain est nul au départ
  i0 := 0; # permet de repérer l'instant d'achat
  j0 := 0; # permet de repérer l'instant de vente
  k := 0;  # permet de repérer l'instant où la cote de
           # l'action est la plus basse
  delta := n; # permet de repérer l'intervalle de temps minimum
  for i from 1 to n-1 do
     if a[i] - a[i0] > g then g := a[i] - a[i0]; j0 := i fi;
     if a[i] - a[k] > g then g := a[i] - a[k]; i0 := k; j0 := i fi;
     if (a[i] - a[k] = g and i-k <= delta) then i0 := k; delta := i-k fi; #
     if a[i] < a[k] then k := i fi; # On change k si la cote
              # à l'instant i est plus basse qu'à l'instant k
  od;
  printf("Instant d'achat i = %d et instant de vente j = %d\n",i0,j0);
  return(g)
end;
```

Question 7 : On écrit une fonction de "`gain_int`" inspirée de la solution de la question 5 qui calcule le gain maximal entre les instants `ti` et `tf` en temps linéaire.

```
gain2 := proc(a::array)
  local i, n, g, g1, g2, gain_int;

  gain_int := proc(a::array, ti, tj::integer)
  local i, i0, n, g, k;
  g := 0;  # Le gain est nul au départ
  i0 := ti; # permet de repérer l'instant d'achat
  k := ti;  # permet de repérer l'instant où la cote d'action est la
  plus basse
  for i from ti to tj do
     if a[i] - a[i0] > g then g := a[i] - a[i0]; fi;  #
     if a[i] - a[k] > g then g := a[i] - a[k]; i0 := k; fi; #
     if a[i] < a[k] then k := i fi; # On change k si la cote
             # à l'instant i est plus basse qu'à l'instant k
  od;
  return(g);
  end; # fin du sous-programme "gain_int"

  # Début du programme principal.
  n := longueur(a); # On récupère la longueur du tableau
  g := 0;  # Le gain total est nul au départ
  g1 := 0; g2 := 0; # les deux gains partiels sont nuls
  for i from 1 to n-1 do
     g1 := gain_int(a,0,i); g2 := gain_int(a,i,n-1);
     if g1+g2 > g then g := g1+g2 fi
  od;
  return(g)
end; #
```

Question 8 :
On écrit une fonction de "`gain_int`" inspirée de la solution de la question 5 qui calcule le gain maximal entre les instants `ti` et `tf` en temps linéaire.

```
gain2 := proc(a::array)
  local i, n, g, r1, r2, transaction1, transaction2, gain_int;

  gain_int := proc(a::array, ti, tj::integer)
  local i, i0, j0, n, g, k;
  g := 0;   # Le gain est nul au départ
  i0 := ti; # permet de repérer l'instant d'achat
  j0 := ti; # permet de repérer l'instant de vente
  k := ti;  # permet de repérer l'instant où la cote
            # de l'action est la plus basse
  for i from ti to tj do
     if a[i] - a[i0] > g then g := a[i] - a[i0]; j0 := i fi;  #
     if a[i] - a[k] >= g then g := a[i] - a[k]; i0 := k; j0 := i fi; #
     if a[i] < a[k] then k := i fi; # On change k si la cote
   od;       # à l'instant i est plus basse qu'à l'instant k
  return([g,i0,j0]); # On retourne le gain, l'instant d'achat
                     # et l'instant de vente
  end; # fin du sous-programme "gain_int"

  # Début du programme principal.
  n := longueur(a); # On récupère la longueur du tableau
  g := 0;  # Le gain total est nul au départ
  transaction1 :=[0,0,0]; transaction2 := [0,0,0];
  for i from 1 to n-1 do
     r1 := gain_int(a,0,i); r2 := gain_int(a,i,n-1);
     if r1[1]+r2[1]> g
        then g := r1[1]+r2[1]; transaction1 := r1; transaction2 := r2 fi
  od;
  printf("La première date d'achat est %d\n",transaction1[2]);
  printf("La première date de vente est %d\n",transaction1[3]);
  printf("La deuxième date d'achat est %d\n",transaction2[2]);
  printf("La deuxième date de vente est %d\n",transaction2[3]);
  return(g)
end;
```