```
> restart;
> allouer:=proc(m,c)
  return Array([seq(c,i=1..m)]);
  end:
> t:=allouer(3,4);
```

$$t := [4, 4, 4]$$

```
> taille:=proc(t)
  return op(2,ArrayDims(t));
  end:
> taille(t);
```

$$3$$

```
> ?
> calculeIndiceMaximum:=proc(tab,a,b)
  local ind,i;
  ind:=a;
  for i from a+1 to b do
          if tab[i]>tab[ind] then
                  ind:=i
          fi;
  od;
  return ind;
  end:
> t:=Array(1..11,[3,2,5,8,1,34,21,6,9,14,8]);n:=11;
```

$$t := \begin{bmatrix} 1..11\ 1\text{-D Array} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

$$n := 11$$

```
> calculeIndiceMaximum(t,1,11);calculeIndiceMaximum(t,1,5);
```

$$6$$

$$4$$

```
> nombrePlusPetit:=proc(tab,a,b,val)
  local nombre,i;
  nombre:=0;
  for i from a to b do
          if tab[i]<=val then
                  nombre:=nombre+1
          fi;
  od;
  return nombre;
  end:
> nombrePlusPetit(t,1,11,5);
```

$$4$$

```
> medianNaif:=proc(tab,a,b)
  local med,i,nbre;
  med:=floor((b-a+1)/2);
```

```
  nbre:=0;
  i:=a;
  while nbre<>med and i<b+1 do
          nbre:=nombrePlusPetit(tab,a,b,tab[i]);
          i:=i+1
  od;
  return i-1;
  end:
> medianNaif(t,1,11);
```



```
> partition:=proc(tab,a,b,indicePivot)
  local i1,i2,p,tabClone,stock,i,fini,n;
  i1:=a;
  i2:=b;
  fini:=1;
  p:=tab[indicePivot];
  n:=taille(tab);
  tabClone:=allouer(n,0);
  for i from 1 to n do
          tabClone[i]:=tab[i]
  od;
  while i1<i2 and fini=1 do
          while i1<=b and tabClone[i1]<p do
                  i1:=i1+1
          od;
          while i2>=a and tabClone[i2]>p do
                  i2:=i2-1
          od;
               if tabClone[i1]<>tabClone[i2] then
                  stock:=tabClone[i2];
                  tabClone[i2]:=tabClone[i1];
                  tabClone[i1]:=stock
          else
                  i:=i1;
                  while tabClone[i]=p and i<i2 do
                          i:=i+1
                  od;
                  if i=i2 then
                          fini:=0
                  else
                          stock:=tabClone[i2];
                          tabClone[i2]:=tabClone[i];
                          tabClone[i]:=stock
                  fi;
          fi;
  od;
```

```
  return [i1,tabClone];
  end:
>
> p:=partition(t,1,11,4);
```

$$p := \left[6, \left[\begin{array}{l} 1..11\ 1\text{-D Array} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{array} \right] \right]$$

```
> convert(p[2],list);
```

$$[3, 2, 5, 1, 6, 8, 8, 21, 9, 14, 34]$$

```
> elementK:=proc(tab,a,b,k)
  local p,i,ttab;
  if k=1 and a=b then
          return tab[a]
  else
          p:=partition(tab,a,b,a);
          i:=p[1];
          ttab:=p[2];
          if i-a+1>k then
                  return elementK(ttab,a,i-1,k)
          elif i-a+1=k then
                  return ttab[i]
          else
                  return elementK(ttab,i+1,b,a-1+k-i)
          fi;
  fi;
  end:

> elementK(t,1,11,6);
```

$$8$$

```
> choixPivot:=proc(tab,a,b)
  local aa,bb,i,taille,ttab;
  if b-a+1<6 then
          taille:=b-a+1;
          return elementK(tab,a,b,floor(taille/2))
  else
          ttab:=allouer(b,0);
          aa:=a;
          bb:=a+4;
          i:=a-1;
          while aa<b+1 do
                  i:=i+1;
                  if bb<b+1 then
                          ttab[i]:=elementK(tab,aa,bb,3)
                  else
                          taille:=b-aa+1;
```

```
    ttab[i]:=elementK(tab,aa,b,floor((1+taille)/2))
                        fi;
                        aa:=aa+5;
                        bb:=bb+5
             od;
             return choixPivot(ttab,a,i)
    fi;
    end:
```

```
> 
> choixPivot(t,1,11);
```

3

```
> ?
> ?
```

7. coupeY :=proc(tabX,tabY)
   local iMed ;
   iMed :=indiceMedian(tabY,1,n) ;
   return tabY[iMed] ;
   end ;
8. demiDroiteMedianeSup :=proc(tabX,tabY,x,y)
   local tabTheta,indPointHaut,i,iMed ;
   tabTheta :=allouer(floor(n/2),0) ;
   indPointHaut :=0 ;
   for i from 1 to n do
       if tabY[i]>y then
           indPointHaut :=indPointHaut+1 ; *(On compte les points au dessus de $L_=$)*
           tabTheta[indPointHaut] :=angle(x,y,tabX[i],tabY[i]) *(On remplit le tableau de leurs angles)*
       fi ;
   od ;
   iMed :=indiceMedian(tabTheta,1,indPointHaut) ;
   return tabTheta(iMed) ;
   end ;
9. verifieAngleSecondeDroite :=proc(tabX,tabY,x,y,theta)
   local l,lGauche,i,phi ;
   l :=0 ;
   lGauche :=0 ;
   for i from 1 to n do
       if tabY[i]<y then
           l :=l+1 ; *(On compte les points au dessous de $L_=$)*
           phi :=angle(x,y,tabX[i],tabY[i]) ;
           if phi<theta then
               lGauche :=lGauche+1 *(Parmi eux, les angles plus petits que θ, ie "en-bas-à-gauche")*
           fi ;
       fi
   od ;
   if lGauche=ceil(l/2) then
       return 0
   elif lGauche>ceil(l/2) then
       return -1
   else
       return 1
   fi ;
   end ;

10. secondeMediane :=proc(tabX,tabY,y)
local succes,a,b,x,theta,score,resultat ;
succes :=-1 ;
a :=xmin ;
b :=xmax ; *(On définit le départ de la dichotomie)*
while succes=-1 do
x :=(a+b)/2 ;
theta :=demiDroiteMedianeSup(tabX,tabY,x,y) ;
score :=verifieAngleSecondeDroite(tabX,tabY,x,y,theta) ;
if score=0 then
succes :=0
elif score=-1 then
b :=x *(On remplace* $[a, b]$ *par* $[a, (a + b)/2]$*)*
else
a :=x *(On remplace* $[a, b]$ *par* $[(a + b)/2, b]$*)*
fi ;
od ;
resultat :=allouer(2,x) ; *(ie* $[x, x]$*)*
resultat[2] :=theta ; *(ce qui donne* $[x, theta]$*)*
return resultat ;
end ;