Correction du Partiel MK1 du 31/10/2006:

Exercice 1:

1) On réalise l'intersection entre l'ensemble des multiples de 6 et l'ensemble des nombres entre 1 et 1000:

```
> restart;
  Mult:={seq(6*i,i=1..1000)} intersect {seq(i,i=1..1000)};
```

$Mult := \{6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72, 78, 84, 90, 96, 102, 108, 114, 120, 126, 132, 138, 144, 150, 156, 162, 168, 174, 180, 186, 192, 198, 204, 210, 216, 222, 228, 234, 240, 246, 252, 258, 264, 270, 276, 282, 288, 294, 300, 306, 312, 318, 324, 330, 336, 342, 348, 354, 360, 366, 372, 378, 384, 390, 396, 402, 408, 414, 420, 426, 432, 438, 444, 450, 456, 462, 468, 474, 480, 486, 492, 498, 504, 510, 516, 522, 528, 534, 540, 546, 552, 558, 564, 570, 576, 582, 588, 594, 600, 606, 612, 618, 624, 630, 636, 642, 648, 654, 660, 666, 672, 678, 684, 690, 696, 702, 708, 714, 720, 726, 732, 738, 744, 750, 756, 762, 768, 774, 780, 786, 792, 798, 804, 810, 816, 822, 828, 834, 840, 846, 852, 858, 864, 870, 876, 882, 888, 894, 900, 906, 912, 918, 924, 930, 936, 942, 948, 954, 960, 966, 972, 978, 984, 990, 996\}$ (1)

On pouvait également demander à Maple de s'arrêter au bon i (la commande *floor* permet d'obtenir la partie entière d'un nombre):

```
> Mult_floor:={seq(6*i,i=1..floor(1000/6))}:
```

2) C'est le même principe; on fait l'intersection des multiples de 6, de 15 et des nombres entre 1 et 100:

```
> Mult_communs:={seq(6*i,i=1..1000)} intersect {seq(15*i,i=1..1000)}
  intersect {seq(i,i=100..1000)};
```

$Mult\_communs := \{120, 150, 180, 210, 240, 270, 300, 330, 360, 390, 420, 450, 480, 510, 540, 570, 600, 630, 660, 690, 720, 750, 780, 810, 840, 870, 900, 930, 960, 990\}$ (2)

Exercice 2:

1) On demande de définir une FONCTION:

```
> f:=x->1/2*(x+9/x);
```

$$f := x \to \frac{1}{2}\, x + \frac{9}{2}\, \frac{1}{x} \tag{3}$$

2) On crée une procédure qui prend n comme variable et on fait un test pour distinguer les différents cas:

```
> u:=proc(n)
  if n<=0 then return erreur
  elif n=1 then return 0.1
  else return f(u(n-1))
  fi;
  end:
```

3)

```
> seq(evalf(u(i)),i=1..10);
```

0.1, 45.05000000, 22.62488901, 11.51134050, 6.146589101, 3.805407910, 3.085231586, 3.001177290, 3.000000231, 3.000000001 (4)

On constate que la suite semble converger vers 3.

Exercice 3:

1) Dans l'aide de student, on trouve la fonction distance qui calcule la distance entre deux points:

```
> with(student);
```

[D, *Diff*, *Doubleint*, *Int*, *Limit*, *Lineint*, *Product*, *Sum*, *Tripleint*, *changevar*, *completesquare*, *distance*, *equate*, *integrand*, *intercept*, *intparts*, *leftbox*, *leftsum*, *makeproc*, *middlebox*, *middlesum*, *midpoint*, *powsubs*, *rightbox*, *rightsum*, *showtangent*, *simpson*, *slope*, *summand*, *trapezoid*] (5)

```
> xA:=1:yA:=2:xB:=3:yB:=-1:
  d:=distance([xA,yA],[xB,yB]);
```

$$d := \sqrt{13} \tag{6}$$

2)

```
> solve({distance([x,y],[xA,yA])=distance([x,y],[xB,yB])},{x,y});
```

$$\left\{x = \frac{5}{4} + \frac{3}{2}\, y,\ y = y\right\} \tag{7}$$

3) Maple nous renvoie un ensemble de solutions indexées par y, il y a une infinité de solutions qui sont les couples (x,y) tels que x=5/4+3y/2. Mathématiquement, l'ensemble des points C équidistants de A et de B est la médiatrice de [AB], la réponse de Maple est donc satisfaisante.

4) On ne demandait que de tracer la droite des points C, ici on a x en fonction de y , on va redemander à Maple les solutions en ne mettant plus que y comme inconnue:

```
> solve({distance([x,y],[xA,yA])=distance([x,y],[xB,yB])},{y});
```

$$\left\{y = \frac{2}{3}\, x - \frac{5}{6}\right\} \tag{8}$$

On obtient bien une équation de droite, on trace maintenant la droite des points C:

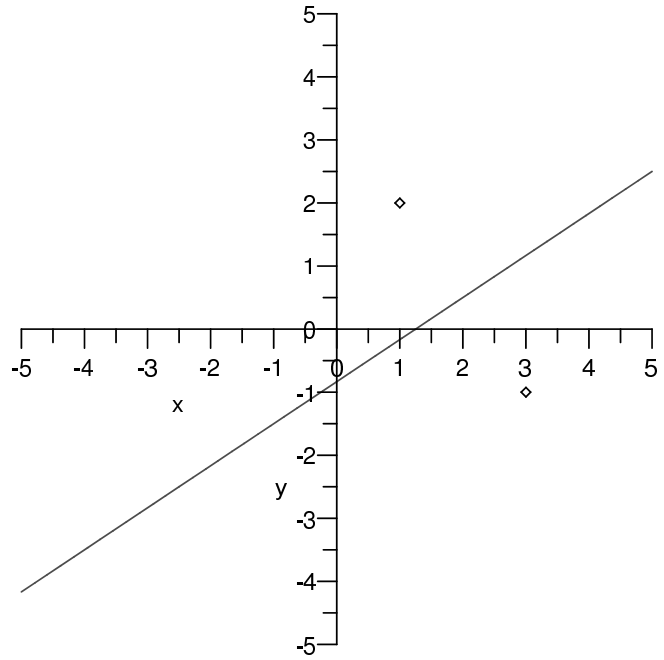
```
> plot(2*x/3-5/6,x=-5..5,y=-5..5):
```

Remarque: Si on veut tracer également les points A et B, on peut faire:

```
> with(plots):
Warning, the name changecoords has been redefined

> AetB:=pointplot({[1,2],[3,-1]}):
  C:=plot(2*x/3-5/6,x=-5..5,y=-5..5):
```

```
display(AetB,C);
```



Exercice 4:

Cette procédure prend un entier n comme argument et renvoie le nombre de ses diviseurs qui sont premiers et strictement inférieurs à n.

La commande irem(n,p) renvoie le reste de la division euclidienne de n par p. Donc, si irem(n,p)=0, cela signifie que p divise n.

```
> proc_inconnue:=proc(n) #cette procedure prend un entier n comme
  argument
  local Res,p; #on declare Res et p comme des variables locales
               #Res sera une liste formee des diviseurs de n
               #p va parcourir les nombres premiers inferieurs à n.
  Res:=[]; #Au debut du programme, Res est la liste vide
  p:=1; #on initialise p a 1
  while p<n #pour chaque p plus petit que n on regarde si p divise
  n:
   do if irem(n,p)=0 #si oui
        then Res:=[op(Res),p]; #on rajoute p à la liste Res
                               #si non on ne fait rien
      fi;
      p:=nextprime(p); #on passe à l'entier p premier suivant
   od;
  nops(Res); #on renvoie le nombre d'elements de la liste Res que
  l'on a créée.
  end:
```

```
> proc_inconnue(20);
```

3 (9)

20 a en effet comme diviseurs premiers stricts 1, 2 et 5 (20=1*2*2*5).