Correction du Partiel MK1 du 03/11/2006:

Exercice 1:

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*] (1)

```
> xA:=1:yA:=2:xB:=5:yB:=-1:
  d:=distance([xA,yA],[xB,yB]);
```

$$d := 5 \tag{2}$$

2)

```
> solve(distance([x,y],[xA,yA])=5,{x,y});
```

$$\left\{x = 1 + \sqrt{21 - y^2 + 4\,y},\ y = y\right\},\ \left\{x = 1 - \sqrt{21 - y^2 + 4\,y},\ y = y\right\} \tag{3}$$

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=1+sqrt(21-y^2+4y) ou x=1-sqrt(21-y^2+4y). Mathématiquement, l'ensemble des points C à distance 5 de A est le cercle de centre A et de rayon 5, la réponse de Maple parait donc satisfaisante.
On trace les points pour s'en convaincre

4) On ne demandait que de tracer l'ensemble 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])=5,{y});
```

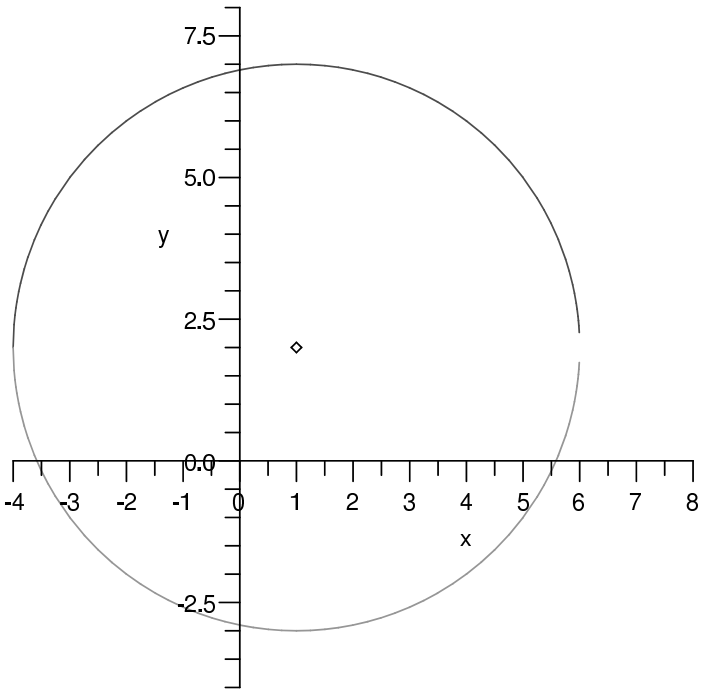
$$\left\{y = 2 + \sqrt{24 - x^2 + 2\,x}\right\},\ \left\{y = 2 - \sqrt{24 - x^2 + 2\,x}\right\} \tag{4}$$

On écrit alors:

```
> plot([2+sqrt(24-x^2+2*x),2-sqrt(24-x^2+2*x)],x=-5..5,y=-5..5):
```

Remarque: Si on veut tracer également le point A , on peut faire:

```
> with(plots):
Warning, the name changecoords has been redefined
> A:=pointplot([1,2]):
  C:=plot([2+sqrt(24-x^2+2*x),2-sqrt(24-x^2+2*x)],x=-4..8,y=-4..8):
  display(A,C);
```



Exercice 2:

1) On réalise l'intersection entre l'ensemble des multiples de 4 et l'ensemble des nombres entre 1 et 500:

```
> restart;
  Mult:={seq(4*i,i=1..500)} intersect {seq(i,i=1..500)};
```

Mult := {4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64, 68, 72, 76, 80, 84, 88, 92, 96, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 144, 148, 152, 156, 160, 164, 168, 172, 176, 180, 184, 188, 192, 196, 200, 204, 208, 212, 216, 220, 224, 228, 232, 236, 240, 244, 248, 252, 256, 260, 264, 268, 272, 276, 280, 284, 288, 292, 296, 300, 304, 308, 312, 316, 320, 324, 328, 332, 336, 340, 344, 348, 352, 356, 360, 364, 368, 372, 376, 380, 384, 388, 392, 396, 400, 404, 408, 412, 416, 420, 424, 428, 432, 436, 440, 444, 448, 452, 456, 460, 464, 468, 472, 476, 480, 484, 488, 492, 496, 500} (5)

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(4*i,i=1..floor(500/4))}:
```

2) C'est le même principe; on fait l'intersection des multiples de 4, de 10 et des nombres entre 100 et 500:

```
> Mult_communs:={seq(4*i,i=1..500)} intersect {seq(10*i,i=1..500)}
  intersect {seq(i,i=100..500)};
```

$$Mult\_communs := \{100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300, 320, 340, 360, 380, 400, 420, 440, 460, 480, 500\} \quad (6)$$

Exercice 3:

La procédure qu'on donne prend un entier n comme argument et renvoie le nombre de ses diviseurs 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 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:=p+1; #on passe à l'entier p suivant
   od;
  nops(Res); #on renvoie le nombre d'elements de la liste Res que
  l'on a créée.
  end:
```

Il fallait donc changer trois choses:

– autoriser p à être égal à n: pour cela on remplace la ligne:

```
> while p<n
```

par:

```
> while p<=n
```

– imposer que p soit premier: on pouvait faire ceci de deux manières:

soit remplacer:

```
> p:=p+1;
```

par:

```
> p:=nextprime(p);
```

soit inclure la condition de primalité dans le test, ie remplacer la ligne:

```
> do if irem(n,p)=0 then...
```

par:

```
> do if (irem(n,p)=0 and isprime(p)=true) then...
```

– changer ce que renvoie la procédure et renvoyer à partir de la liste Res un ensemble, on pouvait pour cela changer la ligne:

```
> nops(Res);
```

en:

```
> {op(Res)};
```

ou:

```
> RETURN({op(Res)}
```

Finalement, on obtient:

```
> diviseur:=proc(n)
  local Res,p;
  Res:=[];
  p:=1;
  while p<=n
  do if irem(n,p)=0 then Res:=[op(Res),p];
     fi;
     p:=nextprime(p);
  od;
  {op(Res)};
  end:
```

```
> diviseur(5);
```

$$\{1, 5\} \quad (7)$$

```
> diviseur(18);
```

$$\{1, 2, 3\} \quad (8)$$

Exercice 4:

1) On demande de définir une FONCTION:

```
> restart;
  f:=x->-x^2+2*x;
```

$$f := x \to -x^2 + 2x \quad (9)$$

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:
```

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

3)

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

0.1, 0.19, 0.3439, 0.56953279, 0.8146979811, 0.9656631616, 0.9988209813,

0.9999986103, 1.000000000, 1.000000000

$$-.1,\ -.21,\ -.4641,\ -1.14358881,\ -3.594972986,\ -20.11377674, \tag{10}$$

$$-444.7915682,\ -1.987291222\ 10^{5},\ -3.949366147\ 10^{10},\ -1.559749296\ 10^{21}$$

On constate que la suite u semble converger vers 1 et la suite v converger vers – infini, ces deux suites ont des comportements très différents alors qu'elle partait "presque de la même condition initiale!