```
> Test :=proc(k, n)
  local c :
  if igcd(k, n) = 1 then c := 1 else c := 0 fi;          #   igcd est la commande maple  pour le pgcd
  c;
  end;

  Test := proc(k, n) local c; if igcd(k, n) = 1 then c := 1 else c := 0 end if; c end proc          (1)
> Test(325, 44); Test(1236, 87);
                                                  1
                                                  0                                                (2)
> Test2 :=proc(k, n)                                    # sans utiliser la commande pgcd
  local a, b, r :
  a := k : b := n :
  r := irem(a, b) :
      #   irem est la commande maple   donnant  le reste de la division euclidienne
  while r > 1 do
      #  on fait l'alogithme d'Euclide et on arrête quand le reste trouvé vaut 1 ou 0,

      # s'il vaut 1 les nombres sont premiers entre eux, s'il vaut 0,les nombres ont un pgcd
      superieur ou égal à 2
  a := b :
  b := r :
  r := irem(a, b) :
  od:
  r;
  end;
Test2 := proc(k, n)                                                                                (3)
  local a, b, r;
  a := k; b := n; r := irem(a, b); while 1 < r do a := b; b := r; r := irem(a, b) end do; r
end proc
> Test2(325, 44); Test2(1236, 87);
                                                  1
                                                  0                                                (4)
> Eucl :=proc(k, n)
      # sans utiliser la commande algorithme d'Euclide avec la commande irem pour rouver le
      pgcd de 2 nombres
  local a, b, r :
  a := k : b := n :
  r := irem(a, b) :
  while r > 0 do
      #  on fait l'alogithme d'Euclide et on arrête quand le reste trouvé  0,   le pgcd est alors
      donné par le quotient b

  a := b :
  b := r :
  r := irem(a, b) :
  od:
  b;
```

```
    end;
Eucl := proc(k, n)                                                    (5)
    local a, b, r;
    a := k; b := n; r := irem(a, b); while 0 < r do a := b; b := r; r := irem(a, b) end do; b
end proc
```

```
> Eucl(32, 12); Eucl(325, 44); Eucl(1236, 87);
                                  4
                                  1
                                  3                                   (6)
```

```
> Card :=proc(n)                          # On compte les inversibles de Z/nZ
    local k, c :
    c := 0;
        # c est un compteur qui compt les inversibles de Z/nZ
    for k from 1 to n − 1 do                # 0=n n'est pas un inversible
    if Test(k, n) = 1 then c := c + 1 fi;
        # quand on trouve un inversible, on incrémente c
    od;
    c;
    end;
Card := proc(n)                                                       (7)
    local k, c;
    c := 0; for k to n − 1 do if Test(k, n) = 1 then c := c + 1 end if end do; c
end proc
```

```
> Card(10); Card(12); Card(29);
                                  4
                                  4
                                 28                                   (8)
```

```
>
> Ord :=proc(k, n)
        # On calcule l'ordre de k dans le groupe des inversibles de Z/nZ
    local i, r, r2 ;
    if Test(k, n) = 0
        then print(ERREUR)
        # on verifie que k est bien inversible dans Z/nZ
        else i := 1 :
            r := irem(k, n) :
        # r est la classe du nombre k dans Z/nZ
            while r ≠ 1 do
        # tant que cette classe est différente de 1 onconinue a calculer les puissances de k dans Z/nZ

        # pour faire le symbole different, on tape: <>
```

```
            r2 := k·r;
            r := irem(r2, n);
            i := i + 1;
            od:
        i;
    fi;
    end;
Ord := proc(k, n)                                                    (9)
    local i, r, r2;
    if Test(k, n) = 0 then
        print(ERREUR)
    else
        i := 1; r := irem(k, n); while r <> 1 do
            r2 := k * r; r := irem(r2, n); i := i + 1
        end do;
        i
    end if
end proc
> Ord(4, 10);
                                    ERREUR                           (10)
> Ord(1, 10); Ord(3, 10); Ord(7, 10); Ord(9, 10);
                                      1
                                      4
                                      4
                                      2                              (11)
>
```