Exercice 1

a)
```
u = 1
for i in range( 50 ) :
    print( u )
    u = 6 * ( u**3 ) - 1
```

b)
```
u = 1
i = 0
while u < 500 :
    u = 6 * ( u**3 ) - 1
    i = i + 1
print( "a l'indice ", i, "u vaut", u )
```

Exercice 2

Lorsque deux solutions sont proposées, celle de droite est un peu plus efficace. Essayer de comprendre pourquoi elles sont correctes et légèrement plus efficaces.

a)
```
def divise0(p, q):
    if q == (q//p) * p :
        return True
    return False
```

```
def divise1(p, q):
    return q%p==0
```

pour 1 million d'appels :

fonction	*divise0*	*divise1*
temps (ms)	*340*	*307*

b)
```
def estpremier0(p):
    if p<2 :
        return 0
    q = 2
    while q*q<=p :
        if p%q==0 :
            return 0
        q = q+1
    return 1
```

```
def estpremier1(p):
    if p==2 :
        return 1
    if p==0 or p==1 or p%2==0 :
        return 0
    q=3
    while q*q<=p :
        if p%q==0 :
            return 0
        q += 2
    return 1
```

pour le premier million de nombres :

fonction	*estpremier0*	*estpremier1*
temps (ms)	*10 830*	*5 591*

c)
```
def nextprime0(N):
    if N<2 :
        return 2
    i = N+1
    while estpremier(i)==0:
        i = i+1
    return i
```

```
def nextprime1(N):
    if N<2 :
        return 2
    i = 2*(N//2)+1     # i devient impair >2
    while estpremier(i)==0:
        i += 2
    return i
```

pour le premier million de nombres (les deux fonctions utilisant `estpremier1`*) :*

fonction	*nextprime0*	*nextprime1*
temps (ms)	*64 991*	*63 887*

d)
```
def phi0(p):
    nbpremier = 0
    for i in range(p+1) :
        if estpremier(i)==1 :
            nbpremier += 1
    return nbpremier
```

```
def phi1(p):
    if p < 2 :
        return 0
    nbpremier = 1  # 2 seul nb premier pair
    for i in range(3,p+1,2) :# nb premiers impairs
        nbpremier += estpremier(i)
    return nbpremier
```

pour les 10 000 premiers nombres (les deux fonctions utilisant `estpremier1`*) :*

fonction	*phi0*	*phi1*
temps (ms)	*49 211*	*40 758*

e)
```
from math import log
def test(epsilon):
    N = 50
    while abs(phi(N)*log(N)/N-1) > epsilon :
        N = N+1
    return N
```

Exercice 3 (avec des piles) :

a)
```
def ordreCroissant(p):
    if taille(p) <= 1 :
        return True
    e1 = depiler(p)
    while taille(p) > 0 :
        e2 = depiler(p)
        if e1 < e2 :
            return False
        e1 = e2
    return True
```

b)
```
def insertion(p, x):
    q = creer_pile()
    while taille(p)>0 and sommet(p)>x :
        empiler(q,depiler(p))
    empiler(p,x)
    while taille(q)>0 :
        empiler(p, depiler(q))
    return p
```

c)
```
def convertStr2piles(phrase):
    p = creer_pile()
    for e in phrase :
        empiler(p, e)
    return p
```

d)
```
def verifPalindrome(p):
    t = taille(p)
    q = creer_pile()
    for i in range(t//2) :
        empiler(q, depiler(p))
    if t%2==1 :
        depiler(p)
    while taille(p)>0 :
        if depiler(p) != depiler(q) :
            return False
    return True
```