Exercice 1

Ecrire un algorithme qui calcule $n!$ à l'aide d'une boucle while. On prouvera la terminaison et la correction de celui-ci

On pose $u_n = n!$. On a $\begin{cases} u_{n+1} = (n+1)u_n \\ u_0 = 1 \end{cases}$.

def factoriel(N):

 u=1

 n=0

 while n<N:

 u=(n+1)*u

 n=n+1

 return u

On introduit deux suites (U_k) et (n_k) tel que $\begin{cases} U_0 = 1, n_0 = 0 \\ U_{k+1} = (n_k + 1)U_k \\ n_{k+1} = n_k + 1 \end{cases}$.

Pour la terminaison : un variant de boucle : $N - n_k$.

$N - n_{k+1} = N - n_k - 1$. $(N - n_k)$ est une suite arithmétique de raison -1 et de premier terme $N \in \mathbb{N}$.

$(N - n_k)$ est une suite d'entiers strictement décroissante.

La condition du while indique que $N - n_k > 0$.

$(N - n_k)$ est un variant de boucle.

Pour la correction : Invariant $U_k = n_k!$

- Avant la boucle : $k = 0, U_0 = 1, n_0! = 0! = 1. U_0 = n_0!$

- Pendant la boucle : supposons que $U_k = n_k!$ pour k fixé.

$$U_{k+1} - n_{k+1}! = (n_k + 1)U_k - (n_k + 1)! = (n_k + 1)n_k! - (n_k + 1)! = 0$$

- Après la boucle : $n_k = N$.
Alors $U_k = N!$
Le programme renvoie $N!$

Exercice 2

Rappelons tout d'abord, la définition de la division entière de deux entiers naturels a et b :

$$\forall a \geq 0, b > 0, a = q.b + r, 0 \leq r < b$$

Pour calculer la division entière, nous allons procéder par soustractions successives de la valeur b de a jusqu'à ce que le résultat de la soustraction soit inférieur à b .

On considère l'algorithme :

```
def divisionEntier(a,b)
    q=0
    r=a
    while r>=b:
        q=q+1
        r=r-b
    return (q,r)
```

1. Exhiber un variant de boucle.

On pose (r_k) et (q_k) les suites tels que

$$\begin{cases} r_0 = a \\ q_0 = 0 \\ q_{k+1} = q_k + 1 \\ r_{k+1} = r_k - b \end{cases}$$

$Mq(r_k)$ est un variant.

(r_k) est une suite arithmétique de raison $-b \in \mathbb{Z}^-$ et de premier terme $a \in \mathbb{N}$.

donc (r_k) est une suite strictement décroissante d'entiers

Or d'après le while $r_k \geq b > 0$.

Donc (r_k) est un variant.

2. Montrer que $a=qb+r$ est un invariant de boucle.

Montrons que $a = q_k b + r_k$.

- Avant la boucle : $q_0 b + r_0 = 0b + a = a$
- Pendant la boucle : supposons $a = q_k b + r_k$, k fixé
 $q_{k+1} b + r_{k+1} = (q_k + 1)b + r_k - b = q_k b + r_k = a$
- Après la boucle : $r_k < b$ et $r_{k-1} \geq b \Leftrightarrow r_{k-1} - b \geq 0 \Leftrightarrow r_k \geq 0$.

Donc $a = q_k b + r_k$ et $0 \leq r_k < b$

Exercice 3

On considère l'algorithme suivant où toutes les variables contiennent des entiers et $n > 1$.

```
def racine_carre(n):  
    inf=1  
    sup=n  
    while inf < sup-1:  
        mid=(inf + sup) // 2  
        if mid*mid <= n:  
            inf=mid  
        else :
```

```

    sup=mid
return inf

```

1. Montrer que l'invariant $inf^2 \leq n < sup^2$ est vrai au départ de la boucle et conservé ensuite.

on pose (inf_k) et (sup_k) définies par

$$\begin{cases} inf_0 = 1, sup_0 = n \\ inf_{k+1} = \begin{cases} \lfloor \frac{inf_k + sup_k}{2} \rfloor & \text{si } \left(\lfloor \frac{inf_k + sup_k}{2} \rfloor\right)^2 \leq n \\ inf_k \end{cases} \\ sup_{k+1} = \begin{cases} sup_k & \text{si } \left(\lfloor \frac{inf_k + sup_k}{2} \rfloor\right)^2 \leq n \\ \lfloor \frac{inf_k + sup_k}{2} \rfloor \end{cases} \end{cases}$$

- Avant la boucle : comme $n \in \mathbb{N}^\bullet$, $1^2 \leq n < n^2$, or $inf_0 = 1$ et $sup_0 = n$, $inf_0^2 \leq n < sup_0^2$
- Pendant la boucle : supposons $inf_k^2 \leq n < sup_k^2$, pour k fixé.
 - Pour $\left(\lfloor \frac{inf_k + sup_k}{2} \rfloor\right)^2 \leq n$: $inf_{k+1}^2 = \left(\lfloor \frac{inf_k + sup_k}{2} \rfloor\right)^2 \leq n$ et $sup_{k+1}^2 = sup_k^2 > n$
 - Pour $\left(\lfloor \frac{inf_k + sup_k}{2} \rfloor\right)^2 > n$: $inf_{k+1}^2 = inf_k^2 \leq n$ et $sup_{k+1}^2 = \left(\lfloor \frac{inf_k + sup_k}{2} \rfloor\right)^2 > n$

2. En déduire que si le programme termine ; il retourne $\lfloor \sqrt{n} \rfloor$.

- Après la boucle : $inf_k \geq sup_k - 1$. On a $inf_k^2 \leq n < sup_k^2$ c'est-à-dire $inf_k < sup_k$. donc $inf_k = sup_k - 1$.
 $inf_k^2 \leq n < sup_k^2 \Leftrightarrow inf_k \leq \sqrt{n} < sup_k = inf_k + 1$. donc $inf_k = \lfloor \sqrt{n} \rfloor$

3. Montrer que la différence $sup - inf$ diminue strictement à chaque étape de boucle. En déduire que le programme termine.

Pour tout $k \geq 0$, $sup_{k+1} - inf_{k+1} = \begin{cases} sup_k - \lfloor \frac{inf_k + sup_k}{2} \rfloor & \text{si } \left(\lfloor \frac{inf_k + sup_k}{2} \rfloor\right)^2 \leq n \\ \lfloor \frac{inf_k + sup_k}{2} \rfloor - inf_k \end{cases}$.

Or $inf_k < sup_k - 1$, $inf_k \leq sup_k - 2$, $inf_k < \lfloor \frac{inf_k + sup_k}{2} \rfloor < sup_k$.

$$\sup_{k+1} - \inf_{k+1} < \sup_k - \inf_k$$

Mq $(\sup_k - \inf_k)$ est un variant de boucle.

Par construction, $\sup_k$ et $\inf_k$ sont des entiers.

$(\sup_k - \inf_k)$ est une suite d'entiers strictement décroissante.

Or $\inf_k < \sup_k - 1 \Leftrightarrow \sup_k - \inf_k > 1$

$(\sup_k - \inf_k)$ est une suite d'entiers positifs strictement décroissante.

Le programme se termine

Exercice 4

Le produit de deux entiers x et y consiste à sommer y fois la valeur x . Toutefois, on peut améliorer cet algorithme rudimentaire en multipliant x par deux et en divisant y par deux chaque fois que sa valeur est paire. Les opérations de multiplication et de division par deux sont des opérations très efficaces en informatique puisqu'elles consistent à décaler de un bit vers la gauche ou vers la droite. Le programme correspondant à cet algorithme est le suivant :

```
def produit(x,y)
    a=x
    b=y
    prod=0
    while b>0:
        while b%2==0:
            b=b//2
            a=2*a
        prod=prod+a
        b=b-1
    return prod
```

1. Déterminer des variants de boucle pour chacune.

2ème boucle

On pose (a_k) et (b_k) définie par
$$\begin{cases} a_0, b_0 \\ b_{k+1} = \frac{b_k}{2} \text{ car } b_k \text{ paire} \\ a_{k+1} = 2a_k \end{cases}$$

(b_k) est un variant de boucle C'est une suite géométrique de raison $\frac{1}{2}$ et de premier terme $b_0 \in \mathbb{N}$.

C'est une suite strictement décroissante et positive. b_k est un entier par construction

1ère boucle.

On définit (α_j) , (β_j) et (prod_j) définies par
$$\begin{cases} \alpha_0 = x, \beta_0 = y, \text{prod}_0 = 0 \\ \alpha_{j+1} = a_\infty \text{ (deuxième while)} \\ \text{prod}_{j+1} = \text{prod}_j + a_\infty \\ \beta_{j+1} = b_\infty - 1 \end{cases}$$

Mq (β_j) est un variant. le while donne $\beta_j > 0$.

Comme (b_k) est un variant, β_{j+1} est un entier $\beta_{j+1} < b_\infty \leq b_0 = \beta_j$.

C'est une suite d'entiers strictement décroissante et positive.

2. Montrer $a b$ constant est un invariant de boucle de 2ème while.

Montrons que $(a_k b_k)$.

On a $a_{k+1} b_{k+1} = 2a_k \frac{b_k}{2} = a_k b_k$.

Doc $a_k b_k$ est constant tout au long de la boucle

3. Montrer que $\text{prod} + ab = xy$ est un invariant du deuxième while.

Mq que $\text{prod}_j + \alpha_j \beta_j = xy$.

- Avant la boucle : $\text{prod}_0 + \alpha_0 \beta_0 = 0 + xy = xy$
- Pendant la boucle : supposons $\text{prod}_j + \alpha_j \beta_j = xy$, pour j fixé
 $\text{prod}_{j+1} + \alpha_{j+1} \beta_{j+1} = \text{prod}_j + a_\infty + a_\infty (b_\infty - 1) = \text{prod}_j + a_\infty b_\infty = \text{prod}_j + a_0 b_0$ car ab constant est l'invariant de la deuxième boucle
 $= \text{prod}_j + \alpha_j \beta_j = xy$
- Après la boucle : $\beta_j = 0$ et $\text{prod}_j + 0 = xy$