CHAPITRE 5 : Les Sous-Programmes

SERIE 1 :

CORRECTION :

Exercice 1 :

```
# sous programme
def verifier_entier_positif(nombre):
    return 10 <= nombre < 100

def saisir_entiers():
    M = int(input("Saisir un entier M (deux chiffres positifs) : "))
    while not verifier_entier_positif(M):
        M = int(input("M invalide. Réessayer : "))

    N = int(input("Saisir un entier N (deux chiffres positifs) : "))
    while not verifier_entier_positif(N):
        N = int(input("N invalide. Réessayer : "))

    return M, N

def afficher_resultat(R):
    print("L'entier formé R est :" ,R)

def former_entier(M, N):
    R = int(str(M)[0] + str(N) + str(M)[1])
    return R

# programme principal
M, N = saisir_entiers()
R = former_entier(M, N)
afficher_resultat(R)
```

Exercice 2 :

```
# sous programme
def verifier_conditions(n, p):
    return n > 0 and p > 0 and n >= p

def saisir_entiers():
    n = int(input("Saisir n (entier naturel positif) : "))
    p = int(input("Saisir p (entier naturel positif, p ≤ n) : "))
    while not verifier_conditions(n, p):
        print("Conditions non respectées. Réessayez.")
        n = int(input("Saisir n : "))
        p = int(input("Saisir p : "))
    return n, p

def factoriel(x):
    result = 1
```

```
    for i in range(1, x + 1):
        result *= i
    return result

def combinaisons(n, p):
    return factoriel(n) // (factoriel(p) * factoriel(n - p))

def afficher_resultat(resultat):
    print(f"Le nombre de combinaisons est : ",resultat)

# programme principal
n, p = saisir_entiers()
resultat = combinaisons(n, p)
afficher_resultat(resultat)
```

Exercice 3 :

```
# sous programme
def verifier_nombre(nombre):
    return 7 <= len(nombre) <= 20 and nombre.isdigit()

def saisir_nombre():
    nombre = input("Saisir un entier contenant entre 7 et 20 chiffres : ")
    while not verifier_nombre(nombre):
        print("Nombre invalide. Réessayez.")
        nombre = input("Saisir un entier valide : ")
    return nombre

def est_divisible_par_11(nombre):
    somme = sum(int(chiffre) if i % 2 == 0 else -int(chiffre) for i, chiffre in enumerate
    return somme % 11 == 0

def afficher_resultat(nombre, divisible):
    if divisible:
        print(nombre ," est divisible par 11.")
    else:
        print(nombre ," n'est pas divisible par 11.")

# programme principal
nombre = saisir_nombre()
divisible = est_divisible_par_11(nombre)
afficher_resultat(nombre, divisible)
```

Exercice 4 :

```
# sous programme
def verifier_chaine(ch):
    return 0 < len(ch) <= 120 and ch.isupper()

def saisir_chaine():
    ch = input("Saisir une chaîne de lettres majuscules : ")
    while not verifier_chaine(ch):
        print("Chaîne invalide. Réessayez.")
        ch = input("Saisir une chaîne valide : ")
    return ch

def crypter_chaine(ch):
```

```
    chr_str = ""
    for char in ch:
        chr_str += str(ord(char))

    chr_reversed = ""
    for i in range(len(chr_str) - 1, -1, -1):
        chr_reversed += chr_str[i]

    resultat = ""
    i = 0
    while i < len(chr_reversed):
        code = chr_reversed[i:i + 2]
        if len(code) == 2:
            resultat += chr(int(code))
        i += 2
    return resultat

def afficher_resultat(resultat):
    print("La chaîne cryptée est : ",resultat)


# programme principal
ch = saisir_chaine()
resultat = crypter_chaine(ch)
afficher_resultat(resultat)
```