Exercice 2 : Institut d'enseignement Néo-moderne (EN)

1.
```
def corrige(cop, corr):
    rep = []
    for i, r in enumerate(cop):
        rep.append(r == corr[i])
    return rep
```

Autre solution :
```
def corrige(cop, corr):
    return [cop[i]==corr[i] for i in range(20)]
```

2.
```
def note(cop, corr):
    note = 0
    for i in range(20):
        if cop[i] == corr[i]:
            note += 1
    return note
```

Autre solution :
```
def note(cop, corr):
    note = 0
    for i, r in enumerate(cop):
        if r == corr[i]:
            note += 1
    return note
```

3.
```
def notes_paquet(p, corr):
    rep = {}
    for cle, val in p.items():
        rep[cle] = note(val, corr)
    return rep
```

Autre solution :
```
def notes_paquet(p, corr):
    return {nom:note(cop, corr) for nom, cop in p.items()}
```

4. On ne peut pas utiliser une liste comme clé d'un dictionnaire car les listes sont mutable.

5. Il faudrait associer à chaque candidats un numéro unique d'identification.
Une autre possibilité moins sûre serait d'utiliser comme identifiant un triplet (Prénom, Nom, Date de naissance).

6. Une table d'exécution permet de répondre :

ligne	a	b	c	d	nom	tmp
6	None	None	None	{}		
8	None	None	None	{}	('Tom','Matt')	None
8	(('Tom','Matt'):6)	None	None	{}	('Lambert','Ginne')	None
8	(('Tom','Matt'):6)	(('Lambert','Ginne'),4)	None	{}	('Carl', 'Roth')	None
8	(('Tom','Matt'):6)	(('Lambert','Ginne'),4)	(('Carl','Roth'):2)	{}	('Kurt', 'Jett')	(('Carl','Roth'):2)
8	(('Tom','Matt'):6)	(('Lambert','Ginne'),4)	(('Kurt','Jett'),4)	{('Carl','Roth'):2}	('Ayet', 'Finzerb')	(('Kurt','Jett'),4)
21	(('Tom','Matt'):6)	(('Lambert','Ginne'),4)	(('Kurt','Jett'),4)	{('Carl','Roth'):2,('Ayet','Finzerb'):3}	('Ayet', 'Finzerb')	(('Kurt','Jett'),4)

La valeur renvoyée est `(a, b, c, d)` soit :
((('Tom','Matt'):6), (('Lambert','Ginne'),4), (('Kurt','Jett'),4), {('Carl','Roth'):2, ('Ayet','Finzerb'):3})

7. La fonction `enigme` permet de déterminer le podium des trois meilleurs candidats.

8. Si le dictionnaire contient strictement moins de 3 candidats, ils sont renvoyés dans l'ordre décroissant des notes, avec un ou plusieurs `None` pour compléter le podium, et un dictionnaire vide en quatrième élément du tuple.

9.
```
def classement(notes):
    classement = []
    while notes != {}:
        a, b, c, notes = enigme(notes)
        for e in [a, b, c]:
            if e:
                classement.append(e)
    return classement
```

10. Utilisation de la recherche par dichotomie.
```
def renote_express2(copcorr):
    gauche = 0
    droite = len(copcorr)
    while droite - gauche > 1:
        milieu = (gauche + droite)//2
        if copcorr[milieu]:
            gauche = milieu
        else:
            droite = milieu
    if copcorr[gauche]:
        return gauche + 1
    else:
        return gauche
```

11. Le coût en temps de la fonction `renote_express` est linéaire (« en O(n) ») puisque le tableau est entièrement parcouru.
Le coût en temps de la fonction `renote_express2` qui applique le principe de la recherche par dichotomie est logarithmique (« en O (log(n)) »), À chaque étape, le nombre d'éléments du tableau considéré est divisé par deux.

12. Il suffit que la fonction `renote_express2` prenne en paramètre la copie `cop` et la correction `corr` et remplacer `copcorr[milieu]` et `copcorr[gauche]` par respectivement `cop[milieu] == corr[milieu]` et `cop[gauche] == corr[gauche]`.