Matthieu Boyer
Planche A

RCW

m personnes ont des comptes dans n banques. On connaît l'argent que chacun a dans chaque banque. Donner un algorithme renvoyant l'argent possédé par la plus riche personne.

Ski

m chaînes de montagne contiennent des montagnes de différentes hauteurs, triées dans l'ordre croissant. Donner un algorithme renvoyant le plus grand dénivelé entre deux montagnes.

Solution: RCW

Solution en Python :

```
def maximumWealth(accounts: List[List[int]]):
        return max([sum(acc) for acc in accounts])
```

Solution: Ski

Solution en Python :

```
def maxdenivele(liste: List[List[int]]):
        smallest = liste[0][0]
        biggest = liste[0][-1]
        max_distance = 0

        for i in range(1, len(liste)):
                arr = liste[i]
                max_distance = max(max_distance, abs(arr[-1] - smallest),
                                   abs(biggest - arr[0]))
                smallest = min(smallest, arr[0])
                biggest = max(biggest, arr[-1])

        return max_distance
```

Matthieu Boyer
Planche B

IPv4

Une adresse IP valide est constituée d'exactement 4 entiers séparés par des points. Chaque entier est sur 8 bits et ne peut commencer par un 0.

1. Quelles sont les valeurs possibles pour les entiers ?
2. Étant donné une chaîne de caractères s ne contenant que des nombres, donner un algorithme qui énumère les IP qu'on peut former en insérant des points dans s.

Solution: IPv4

La solution ci-dessous est techniquement en C++ mais n'utilise aucune des fonctionnalités du langage à des fins *intéressantes* autres que stocker les résultats dans une liste.

```
class Solution \{
public:
        bool isValid(string &s, int indx, int len)\{
                if (indx + len > s.size()) return false;
                string _found = s.substr(indx, len);
                if (_found.size() > 1 && _found[0] == '0') return false;
                int num = stoi(_found);
                if (num > 255) return false;
                return true;
        \}
        void recur(string &s, int indx, int used, vector<string> &res, string curr)\{
                        curr.pop_back();
                        res.push_back(curr);
                        return;
                \}
                if (indx >= s.size() ||used >= 4)\{
                        return;
                \}
                if (isValid(s, indx, 1))\{
                        recur(s, indx + 1, used + 1, res, curr + s.substr(indx,1) + '.');
                \}
                if (isValid(s, indx, 2))\{
                        recur(s, indx + 2, used + 1, res, curr + s.substr(indx,2) + '.');
                \}
                if (isValid(s, indx, 3))\{
                        recur(s, indx + 3, used + 1, res, curr + s.substr(indx,3) + '.');
                \}
        \}

        vector<string> restoreIpAddresses(string s) \{
                if (s.size() > 12) return \{\};
                vector<string> res;
                string curr = '';
                recur(s, 0, 0, res, curr);
                return res;
        \}
\};
```

Matthieu Boyer
Planche C

Bourse.

Vous êtes analyste financier. On vous transmet la liste des valeurs de l'action des m dernières ouvertures du marché. On vous autorise à effectuer au plus k transactions (achats et ventes). Il n'est pas possible de faire plusieurs transactions en même temps.

Solution: Bourse.

```
def maxProfit(k, prices):
        if k == 0: return 0
        \# dp[k][0] = min coût k transactions
        \# dp[k][1] = max profit k transactions
        dp = [[len(k), 0] for _ in range(k + 1)]
        for price in prices:
                for i in range(1, k + 1):
                        dp[i][0] = min(dp[i][0], price - dp[i - 1][1])
                        dp[i][1] = max(dp[i][1], price - dp[i][0])
        return dp[k][1]
```