HEIG VD

# ASD - Labo 2
# Puzzle Impossible

# Principe du jeu

- 9 pièces carrées uniques (numérotées de 1 à 9)
- Chaque pièce a 4 demi-figurines (1 de chaque)
- 1 demi-figurine qui ne coïncide pas (arrosoir inversé)
- 4 figurines (dame, fille/princesse, gâteau, arrosoir) avec deux moitiés (haut et bas, gauche et droite)
- 2 pièces coïncident si les deux parties forment une figurine en entier (haut et bas, gauche et droite)

| Figurine | Gauche/Haut | Droite/Bas |
|---|---|---|
| Gâteau | 3 | 6 |
| Arrosoir | 4 | 4 (+1 inversé) |
| Fille/Princesse | 5 | 4 |
| Dame | 6 | 3 |

# Données : Pieces.h

```
enum AttachementType {
  FILLE_HAUT,
  FILLE_BAS,
  DAME_HAUT,
  DAME_BAS,
  ARROSOIR_GAUCHE,
  ARROSOIR_DROIT,
  GATEAU_GAUCHE,
  GATEAU_DROIT,
  ARROSOIR_INVERSE,
  NONE };

using Piece = std::array<AttachementType,4>;

using Pieces = std::vector<Piece>;
```

# Données : Pieces.cpp

```
const Pieces PIECES = {
   { DAME_HAUT, GATEAU_DROIT, ARROSOIR_GAUCHE, FILLE_HAUT },
   { DAME_BAS, ARROSOIR_GAUCHE, FILLE_HAUT, GATEAU_DROIT },
   { FILLE_BAS, GATEAU_GAUCHE, DAME_HAUT, ARROSOIR_DROIT },
   { ARROSOIR_DROIT, GATEAU_GAUCHE, DAME_HAUT, FILLE_HAUT},
   { FILLE_BAS, DAME_HAUT, ARROSOIR_DROIT, GATEAU_DROIT },
   { DAME_BAS, GATEAU_GAUCHE, FILLE_HAUT, ARROSOIR_DROIT},
   { FILLE_BAS, ARROSOIR_GAUCHE, DAME_HAUT, GATEAU_DROIT },
   { DAME_BAS, ARROSOIR_GAUCHE, GATEAU_DROIT, FILLE_HAUT },
   { ARROSOIR_INVERSE, DAME_HAUT, GATEAU_DROIT, FILLE_BAS },
};
```

# Rotations


```
1:  { DAME_HAUT, GATEAU_DROIT, ARROSOIR_GAUCHE, FILLE_HAUT }

1a: { DAME_HAUT, GATEAU_DROIT, ARROSOIR_GAUCHE, FILLE_HAUT }
1b: { GATEAU_DROIT, ARROSOIR_GAUCHE, FILLE_HAUT, DAME_HAUT }
1c: { ARROSOIR_GAUCHE, FILLE_HAUT, DAME_HAUT, GATEAU_DROIT }
1d: { FILLE_HAUT, DAME_HAUT, GATEAU_DROIT, ARROSOIR_GAUCHE }
```




# Principe de la solution

- Ecrire un algorithme récursif capable de générer toutes les permutations des 9 pièces et toutes les rotations de ces pièces
- La fonction récursive essaye de remplir une des 9 positions
  - Elle boucle sur toutes les pièces restantes / les 4 rotations et essaye de les placer à cette position
  - Si placer la pièce/rotation est compatible avec les autres pièces placées jusque là, elle appelle récursivement pour la position suivante
  - Jusqu’à ce que les 9 positions soient remplies

# A faire

- Forker le repository https://github.com/ASD-HEIGVD/ASD-L2-Recursivite
- Ecrire un programme générant toutes les solutions au format permettant de les tester sur https://asd1-heigvd.github.io/ASD1-Labs/puzzle/
- Rédiger un rapport qui considère de manière critique le slogan « des millions de possibilités et une seule solution » incluant au moins les 3 nombres suivants
  - Nombre total de possibilités
  - Nombre de solutions (et lesquelles)
  - Nombre d'appels à votre fonction récursive en ne parcourant pas les branches inutiles de l'arbre des appels