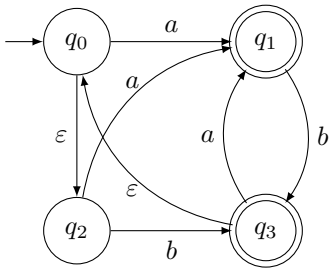
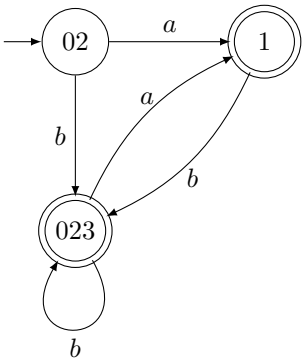
Exercice 1

1. C'est un langage appartenant au plus petit ensemble contenant le langage vide, les langages réduits à un singleton d'un mot d'une seule lettre et stable par union, concaténation et étoile.
2. (a) $L_1 = a^*ba^*$ est régulier
   (b) L_2 n'est pas régulier. S'il l'était, soit n sa longueur de pompage. On pose $u = a^nba^n \in L_2$. Soit alors $u = xyz$ tels que $|xy| \leqslant n$ et $y \neq \varepsilon$. On en déduit que $y = a^k$, $k \in [\![1, n]\!]$, puis que $xy^2z = a^{n+k}ba^n \notin L_2$, ce qui est contradictoire d'après le lemme de pompage.
   (c) On remarque que $L_3 = L_1 \setminus L_2$. S'il était régulier, alors $L_2 = L_1 \setminus L_3$ le serait aussi, ce qui est absurde.
   (d) $L_4 = (aa)^*(aba \mid b)(aa)^*$ est régulier.
3. 

   (a) On construit l'automate des parties directement. C'est le même algorithme que pour un automate des parties d'un AFND, sauf qu'il faut appliquer des ε-clôtures pour les transitions, et pour déterminer la partie initiale.

Partie	a	b
02	1	023
1	$\emptyset$	023
023	1	023

   Soit :



   (b) On pourrait appliquer l'algorithme d'élimination des états, mais on peut simplement utiliser la question suivante pour trouver $(b \mid ab)^*(a \mid b)$.
   (c) Il s'agit des mots non vides ne contenant pas aa comme facteur.

Exercice 2

1. On remarque que pour toute case du tableau, la case à droite et en dessous ont la même valeur. Ainsi, tous les chemins récoltent le même nombre de fleurs, à savoir 11.
2. On a $n(i, j) = \max(n(i-1, j), n(i, j-1)) + C[i, j]$, où $C[i, j]$ est la matrice correspondant au champ.

   On peut poser $n(i, j) = 0$ si $i < 0$ ou $j < 0$ comme cas de base. La question et la remarque qui

suivent suggèrent une implémentation récursive naïve.

```
int recolte(int champ[m][n], int i, int j){
    if (i < 0 || j < 0) return 0;
    return champ[i][j] + max(recolte(champ, i - 1, j), recolte(champ, i, j - 1));
}
```

3. Voici la liste des appels récursifs qui peuvent mener à un appel pour $(1, 1)$:
   - un appel pour $(3, 3)$;
   - un appel pour $(2, 3)$ et un pour $(3, 2)$;
   - un appel pour $(1, 3)$, un pour $(3, 1)$ et deux pour $(2, 2)$;
   - trois appels pour $(1, 2)$ et trois pour $(2, 1)$;
   - soit 6 appels pour $(1, 1)$.
4. On peut remplir par valeurs croissantes de $i + j$, donc par contre-diagonales croissantes.
5. On obtient :

```
int recolte_iterative(int champ[m][n], int i, int j,int fleurs[m][n]){
    for (int k=0; k<m + n; k++){
        for (int p=0; p<=k; p++){
            int q = k - p;
            int f1 = (p>0)?fleurs[p - 1][q]:0;
            int f2 = (q>0)?fleurs[p][q - 1]:0;
            fleurs[p][q] = champ[p][q] + max(f1, f2);
        }
    }
    return fleurs[i][j];
}
```

6. On peut avoir :

```
void deplacements(int fleurs[m][n], int i, int j){
    int p = 0;
    int q = 0;
    while (p < i || q < j){
        if (p == i) {
            printf("droite, ");
            q++;
        } else if (q == j) {
            printf("bas, ");
            p++;
        } else if (fleurs[p + 1][q] > fleurs[p][q + 1]){
            printf("bas, ");
            p++;
        } else {
            printf("droite, ");
            q++;
        }
    }
    printf("\n");
}
```

7. Juste avant le `return`.