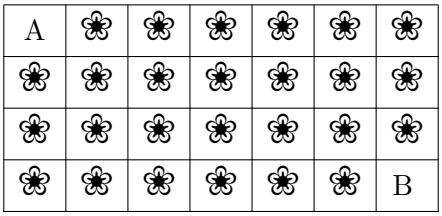
Exercice 6 *Récolte dynamique de fleurs (type B)*

Consignes : Cet énoncé est accompagné d'un code compagnon en C bouquet_enonce.c *fournissant certaines des fonctions mentionnées dans l'énoncé : il est à compléter en y implémentant les fonctions demandées. La ligne de compilation* gcc -o main.exe *.c -lm *vous permet de créer un exécutable* main.exe *à partir du ou des fichiers C fournis. Vous pouvez également utiliser l'utilitaire* make. *En ligne de commande, il suffit de taper* make. *Dans les deux cas, si la compilation réussit, le programme peut être exécuté avec la commande* ./main.exe. *Si vous désirez forcer la compilation de tous les fichiers, vous pouvez au préalable nettoyer le répertoire en faisant* make clean *et relancer* make.

Une petite fille se trouve en haut à gauche (case A) d'un champ modélisé par un tableau rectangulaire de taille $m \times n$ et doit se rendre dans la case B en bas à droite du champ où réside sa grand-mère (figure ci-dessous).



Chaque case du tableau, *y compris les cases A et B*, contient un certain nombre de fleurs. La petite fille, qui connaît depuis sa position initiale le nombre de fleurs de chaque case, doit se déplacer vers B de case en case, les seuls mouvements autorisés étant vers le bas ou vers la droite. À chaque déplacement, elle récolte les fleurs de la case atteinte. L'objectif pour elle est alors de faire le bouquet avec le plus de fleurs possible lors de son déplacement pour l'offrir à sa grand-mère.

1. On considère le champ suivant :

0 (A)	1	2	3
1	2	3	4
2	3	4	0
3	4	0	1 (B)

   Donner le nombre maximal de fleurs cueillies par la petite fille.

2. On note $n(i,j)$ le nombre maximum de fleurs que la petite fille peut récolter en se déplaçant de A à la case (i,j). Exprimer $n(i,j)$ en fonction de $n(i-1,j)$ et $n(i,j-1)$. En déduire une fonction récursive de prototype `int recolte(int champ[m][n], int i, int j)` qui, étant données les coordonnées `i`, `j` d'une case, calcule le nombre maximum de fleurs cueillies par la petite fille de A à la case (`i`, `j`).

3. On suppose $m = n = 4$ et on effectue donc un appel à `recolte(champ,3,3)` pour résoudre le problème posé. Donner le nombre de fois où votre fonction calcule le nombre de fleurs maximum cueillies dans la case $(1,1)$ (deuxième case de la diagonale).

D'une manière générale, le nombre d'appels à la fonction récursive est important. On a donc intérêt à transformer l'algorithme récursif en algorithme dynamique. On propose de déclarer dans le programme principal un tableau `fleurs` dont la case (i,j) est destinée à contenir la récolte maximale que la petite fille peut obtenir en cheminant de A vers la case (i,j).

4. Dans quel ordre remplir le tableau `fleurs` de sorte à éviter de recalculer une valeur ?

5. Écrire une fonction de prototype `int recolte_iterative(int champ[m][n], int i, int j, int fleurs[m][n])` qui calcule, stocke dans `fleurs[i][j]` et retourne la cueillette maximale obtenue en parcourant le champ de A à la case (`i`, `j`).

La fonction `recolte_iterative` permet de déterminer la cueillette maximale en (`i`, `j`) mais ne précise pas le chemin parcouru pour l'obtenir.

6. Écrire la fonction de prototype `void deplacements(int fleurs[m][n], int i, int j)` qui affiche la suite des déplacements effectués par la petite fille sur un chemin permettant de récolter le nombre maximum de fleurs entre (0,0) et (`i`, `j`).
7. Insérer un appel de `deplacements` dans la fonction `recolte_iterative` pour afficher le chemin parcouru.

Proposition de corrigé

1. On trouve 11 fleurs.
2. La récolte en une case dépend récursivement de celle de la case du haut et de celle de la case de gauche. La formule générique est donc à adapter lorsqu'on se trouve sur le bord gauche ou le bord haut du champ.

```
int recolte(int champ[m][n], int i, int j){
  /* Cas de base */
  if ( (i == 0) && (j == 0) )
   return champ[0][0];
  if (i == 0)
    return champ[0][j] + recolte(champ,0, j - 1);
  if (j == 0)
    return champ[i][0] + recolte(champ,i - 1, 0
  /* Cas général */
  return champ[i][j] + max(recolte(champ,i - 1, j),recolte(champ,i, j - 1));
}
```

3. On trouve 6 appels à `recolte`.
4. On calcule d'abord les valeurs des cases sur les bords haut et gauche puis on propage soit en remplissant les lignes de gauche à droite, soit en remplissant les colonnes de haut en bas.
5. Il s'agit d'une traduction des questions 2 et 4. Le code ci-dessous corrige aussi la question 7.

```
int recolte_iterative(int champ[m][n], int i, int j,int fleurs[m][n]){
  int x, y;
  fleurs[0][0] = champ[0][0];
  /* Bord haut */
  for (x = 1; x <= i; x++) {
    fleurs[x][0] = champ[x][0] + fleurs[x - 1][0];
  }
  /* Bord gauche */
  for (y = 1; y <= j; y++) {
    fleurs[0][y] = champ[0][y] + fleurs[0][y - 1];
  }
  /* Autres cases */
  for (y = 1; y <= j; y++) {
    for (x = 1; x <= i; x++) {
      fleurs[x][y] = champ[x][y] + max(fleurs[x - 1][y], fleurs[x][y - 1]);
    }
  }

  deplacements(fleurs,i, j);
  return fleurs[i][j];
}
```