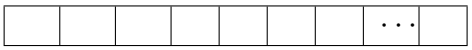
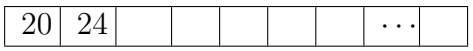
TP5 - à rendre dans une semaine
Les Tableaux

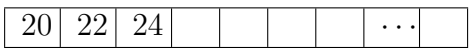
1 Tri de tableau en ligne

On désire construire un tableau d'au plus MAX entiers triés à partir d'un dialogue avec l'utilisateur. La particularité de ce tri est qu'il va se faire au fur et à mesure de la saisie des valeurs.

Exemple, on part d'un tableau vide : 

l'utilisateur rentre 24, le tableau devient : 

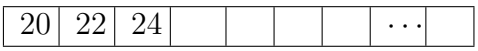
l'utilisateur rentre 20, le tableau devient : 

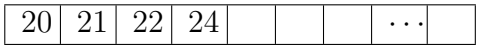
l'utilisateur rentre 22, le tableau devient : 

etc.

L'algorithme consiste pour chaque nouvel entier entré par l'utilisateur à rechercher dans le tableau partiellement rempli, la position que doit prendre le nouvel élément puis à l'insèrer dans le tableau à la position spécifiée.
Exemple

pour le tableau :  et l'entier 21, la position d'insertion est 1.

insérer l'entier 21 **à la position** 1 **dans le tableau :** 

modifie le tableau de la façon suivante : 

1. Définir quelques jeux de données qui vous semblent pertinents pour tester le tri.
2. Ecrire une fonction $affiche Tab(tab, n)$ qui affiche un tableau de n éléments. Chaque élément sera séparé par une tabulation.
3. Écrire la fonction principale $main()$ qui invite l'utilisateur à saisir les nombres entiers, construit et affiche le tableau trié au fur et à mesure de la lecture des nombres. On veillera à ne pas dépasser la capacité maximale du tableau.
4. Vérifier que votre programme s'exécute correctement avec les jeux de données de la première question

2 Le jeu de la vie

Le jeu de la vie fut inventé par John Horton Conway en 1970, décrivant le développement, le déclin et les altérations d'une colonie de micro-organismes. Le but de cet exercice est d'en programmer une version simplifiée. Nous disposons d'une grille, dans laquelle chaque case peut contenir une cellule au plus. Les cases peuvent prendre les valeurs 0 (cellule morte ou pas de cellule) ou 1 (cellule vivante). À chaque étape, on calcule la nouvelle valeur de chacune des cases en fonction des valeurs des cases voisines. Une case a 8 cases voisines : haut, bas, droite, gauche et les 4 diagonales.

— La valeur d'une case de valeur 1 passe à 0 si elle possède 0 ou 1 ou 4 voisins ou plus, à 1 (elle meurt isolée ou étouffée).
— La valeur d'une case de valeur 0 passe à 1 si elle possède exactement 3 voisins à 1 (elle renaît).
— Elle ne change pas dans les autres cas.

Pour éviter les cas particuliers des cases se trouvant en périphérie de la grille, nous considérerons que cette dernière est stockée dans un tableau comprenant une bordure remplie de zéros (0) qui n'évoluent pas.
Exemple.
La figure ci-dessous représente une grille 4x4 avec sa bordure d'où la grille 6x6. La case à l'intersection de la 3ème ligne et de la 3ème colonne (grisée foncé) a trois voisins à 1 et un voisin à 0.
Elle passera donc à 1 à la prochaine étape.

0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	0	0	1	0
0	0	1	1	0	0
0	0	0	0	0	0

1. Ecrire une fonction $affichGrille(grille)$
2. Ecrire une fonction $nbVoisins1(grille, ligne, colonne)$ qui retourne le nombre de voisins à 1 pour la case [ligne,colonne] de la grille.
3. En utilisant la fonction précédente, écrire la fonction $evolutionCellule(grille, ligne, colonne)$ qui retourne la valeur que devra prendre la case [ligne,colonne] à l'étape suivante.
4. Ecrire la fonction $changeEtat(grille)$ qui permet de faire évoluer l'ensemble de la grille d'une étape vers une autre. Attention, on devra utiliser un deuxième tableau dans la fonction qui permettra de stocker temporairement le nouvel état.
5. Ecrire la fonction principale $main()$ qui demande à l'utilisateur le nombre d'étapes qu'il souhaite, puis affichera chacune des étapes de l'évolution. La grille sera initialisée comme dans le ficher init.c mis à disposition sur l'ENT.
6. Vérifier que le résultat est bien celui attendu.
7. Tester votre programme avec d'autres initialisations de grille.
8. Question bonus : Ecrire une fonction qui initialise la grille avec une valeur (0 ou 1) au hasard pour chacune des cases . On utilisera la fonction $rand()$ vue en cours ou TD. Les bordures doivent être initialisées à 0.

3 Travail à rendre

Chaque binôme doit déposer sur le moodle une archive (.zip) contenant les fichiers avec le code commenté (.c) et un compte-rendu décrivant les tests effectués, les limites de votre programme et les problèmes éventuellement rencontrés. La date limite de dépôt est date du TP + 7 jours.