# LIFAPI – Partie A - Algorithmique

**Contrôle Continu Terminal** (Durée totale : 2h)
Lundi 9 janvier 2023

NOM :

……

PRENOM :

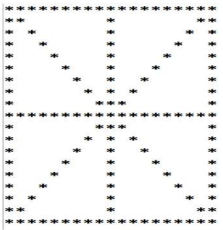
……

Numéro Etudiant :

……

***Recommandations :*** *Les documents, calculatrice, téléphone portable sont interdits. La qualité de l'écriture et de la présentation seront prises en compte dans la note finale. Vous veillerez à* ***respecter*** *les notations et les règles d'écriture des algorithmes vues en cours et en TD. Un soin tout particulier devra être apporté à l'écriture des entêtes des différents sous-programmes.*

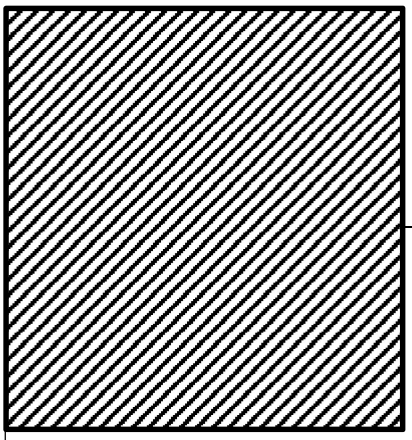
## Exercice I : God save the king



Nous allons écrire un programme permettant de dessiner et d'afficher à l'écran le motif ci-contre (ressemblant au drapeau britannique). Pour des raisons pratiques et esthétiques, la hauteur du motif sera **strictement supérieure à 15** et nécessairement **impaire**. Le caractère utilisé sera choisi par l'utilisateur dans le programme principal.

1- Ecrire l'algorithme d'une **fonction** `saisie_valeur` qui demande à l'utilisateur un entier **strictement supérieur à 15 et impair** et retourne la valeur choisie. La saisie sera recommencée tant que la valeur proposée n'est pas strictement supérieure à 15 et impaire.

.



2- Ecrire l'algorithme du sous-programme `motif` qui prend en paramètres la hauteur `h` du dessin ainsi que le caractère `c` utilisé et qui affiche le motif voulu. Dans l'exemple `h = 19` et `c = '*'`. On pourra utiliser l'instruction `afficher (saut de ligne)` pour passer à la ligne suivante.

3- En utilisant les sous-programmes précédents, écrire l'algorithme du programme principal qui permet de saisir une valeur impaire strictement supérieure à 15, un caractère et affiche le motif correspondant.

## Exercice II : Minuscules ou majuscules ?

Nous allons écrire quelques outils de manipulation de chaines de caractères. Dans cet exercice, vous pourrez utiliser la fonction `alea()` qui retourne un entier compris entre 0 et une constante `MAX = 32767` (fonction équivalente à la fonction rand() du C++), ainsi que la fonction `longueur` qui retourne la longueur d'une chaine de caractères passée en paramètre. Aucun programme principal ne sera demandé dans cet exercice.

1- Ecrire l'algorithme d'un sous-programme `genere_chaine` qui construit et "retourne" une chaine de caractères de longueur `lg`, telle que tous les caractères d'indices **pairs** seront écrits en **minuscules** et tous les caractères d'indices **impairs** seront écrits en **majuscules**. La longueur de la chaine sera passée en paramètres. Les caractères seront générés de manière aléatoire uniquement parmi les caractères alphabétiques. On suppose qu'une constante `CHMAX = 64` a été définie au préalable.
Exemple de chaine générée avec `lg = 7` ➔ `zMsVaUq`

2- Ecrire l'algorithme d'un sous-programme `inverse_casse` qui, à partir d'une chaine de caractères passée en paramètre, construit une nouvelle chaine dans laquelle toutes les minuscules sont transformées en majuscules et vice-versa.
Exemple sur la chaine précédente ➔ `ZmSvAuQ`

3- Ecrire l'algorithme d'une **fonction booléenne** `compare` qui retourne vrai si les deux chaines de caractères passées en paramètres sont identiques **quelle que soit la casse** (majuscule ou minuscule), faux sinon. L'utilisation d'un équivalent de `strcmp` est interdite.
Par exemple `compare("zMsVaUq", "ZmSvAuQ")` retournera vrai mais `compare("HeLlO", "BoNjOuR")` renverra faux.