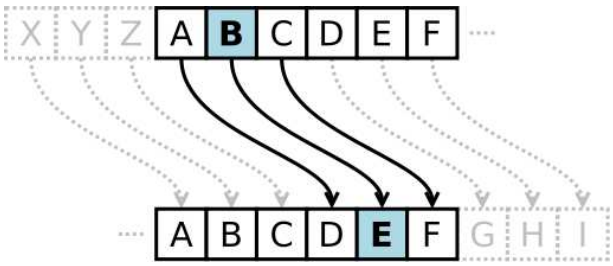
Nom :	Note : ..... / 20
Prénom :	

❑ **Exercice 1** : *Code de César*

L'une des plus ancienne méthodes de chiffrement est le code de César qui consiste simplement à décaler chaque lettre d'une distance fixe dans l'alphabet. Par exemple si cette distance (appelée clé de chiffrement) est 3, la lettre A devient D, B devient E, et ainsi de suite. Pour les dernières lettres on reprend au début de l'alphabet. Par exemple toujours avec une distance de 3, X devient A, Y devient B et Z devient C. Ce fonctionnement est illustré ci-dessous :



Le but de l'exercice est de créer un exécutable `cesar.exe` qui prend en argument une chaine de caractères et une clé de chiffrement et affiche le texte chiffré correspondant. On ne décalera que les lettres (majuscules ou mininuscules) et on laissera les autres caractères intacts. Par exemple :

```
./cesar.exe "La MP2I c'est trop bien !" 7
Sh TW2P j'lza ayvw iplu !
./cesar.exe "Sh TW2P j'lza ayvw iplu !" -7
La MP2I c'est trop bien !
```

🛟 On rappelle qu'en C, un caractère est represénté par son code ASCII et donc une comparaison comme `car >= 'A'` où `car` est de type `char` est tout a fait légitime car on compare en fait les codes ASCII.

1. Ecrire une fonction `chiffre_caractere` qui prend en argument un caractère `car` et un entier `cle` et renvoie un le caractère `car` chiffré avec le décalage `cle` tel que décrit ci-dessus.
   Par exemple `chiffre_caractere('A',3)` doit renvoyer `'D'`. On rappelle qu'on laisse intact les caractères autres que les lettres majuscules ou minuscules, donc par exemple `chiffre_caractere('!',3)` renvoie `'!'`.
   🛟 En cas de difficultés sur cette première question, pour ne pas rester bloqué, poursuivre l'exercice en utilisant à la place une fonction qui renvoie simplement le caractère passé en argument.

```c
char chiffre_car(char car, int cle)
{
    cle = (cle % 26);
    if (cle < 0)
    {
        cle = cle + 26;
    }
    if ('A' <= car && car <= 'Z')
    {
        return 'A' + ((car - 'A') + cle) % 26;
    }
    if ('a' <= car && car <= 'z')
    {
        return 'a' + ((car - 'a') + cle) % 26;
    }
    return car;
}
```

2. Ecrire une fonction `chiffre_chaine` qui prend en argument une chaine de caractères et et un entier `cle` et renvoie la chaine chiffrée avec le code de César de décalage `cle`.

⚠ On rappelle qu'en C, une chaine de caractères est un tableau de caractères équipé d'une sentinelle marquant la fin de la chaine. Ainsi la chaine `"MP2I"` est de longueur 4 mais occupe 5 caractères en mémoire (le dernier est `'\0'`).

```c
char *chiffre_chaine(char chaine[], int cle)
{
    int size = strlen(chaine);
    char *resultat = malloc(sizeof(char) * (size + 1));
    for (int i = 0; i < size; i++)
    {
        resultat[i] = chiffre_car(chaine[i], cle);
    }
    resultat[size] = '\0';
    return resultat;
}
```

3. Ecrire une fonction `main` qui prend deux arguments en ligne de commande (la chaine à chiffrer et la clé) et affiche la chaine chiffrée dans le terminal. Si on donne à l'exécutable moins de deux arguments, un message d'erreur s'affiche. Par exemple :

```
./cesar.exe "Hello" 11
Spwwz
./cesar.exe "Trop cool"
Erreur, il faut donner une chaine et une clé de chiffrement
```

⛒ On rappelle que l'entier fourni en ligne de commande est une chaine de caractères et que pour le convertir en `int`, on utilise la fonction `atoi` disponible dans `<stdlib.h>`

```c
int main(int argc, char *argv[])
{
    if (argc != 3)
    {
        printf("Il faut donner en argument une chaine de caractères et une clé de
↪   chiffrement\n");
    }
    else
    {
        char *res;
        res = chiffre_chaine(argv[1], atoi(argv[2]));
        printf("%s\n", res);
        free(res);
    }
}
```

4. Ecrire un nouvel executable `cesar_fichier` qui prend en argument un nom de fichier (à la place de la chaine de caractères) ainsi qu'un entier `cle`. C'est le contenu du fichier (limité à 255 caractères) qui est alors chiffré avec le decalage `cle` donné, le résultat est écrit dans le terminal comme précédemment.
⛒ On pourra utiliser la fonction `fgetc` qui lit un caractère sur le flux de lecture donné en argument.

```c
int main(int argc, char *argv[])
{
    if (argc != 3)
    {
        printf("Il faut donner en argument un nom de fichier et une clé de chiffrement\n");
    }
    else
    {
        char *res, car;
        char *data = malloc(sizeof(char) * 255);
        FILE *lecteur = fopen(argv[1], "r");
        int nb_char = 0;
        while (nb_char < 255 && (car = fgetc(lecteur)) != EOF)
        {
            data[nb_char] = car;
            nb_char++;
        }
        data[nb_char] = '\0';
        fclose(lecteur);
        res = chiffre_chaine(data, atoi(argv[2]));
        printf("%s\n", res);
        free(res);
        free(data);
    }
}
```