Séance 1 : Mise en place de la grille et déplacement basique

Objectif : Implémenter la grille et les mouvements de base du robot.

1) Initialiser la grille 2D :
- Créer une matrice (tableau 2D) qui représente la grille. Chaque case peut être un espace vide (' '), un obstacle ('X'), ou le robot ('R').
- Positionner le robot au centre de la grille ou dans un coin spécifique.
- Placer des obstacles à des endroits fixes ou les générer aléatoirement. Utiliser un symbole comme 'X' pour représenter un obstacle.

3) Commandes de déplacement :
- Implémenter des fonctions pour que le robot se déplace :
void haut(), void Gauche(),void Droite(), void bas()
- Avant de faire avancer le robot, vérifier si la case vers laquelle il se dirige contient un obstacle. Si un obstacle est détecté, empêcher le robot de se déplacer et afficher un message d'erreur ou ignorer la commande.

4) Affichage de la grille :
- Mettre à jour la grille après chaque mouvement pour que l'utilisateur puisse voir la nouvelle position du robot.
- Afficher la grille dans la console après chaque commande pour visualiser la progression.

- Correction

Correction

```c
#include <stdio.h>
#define ROWS 10 // Nombre de lignes de la grille
#define COLS 10 // Nombre de colonnes de la grille

void initGrille(char grille[ROWS][COLS], int robotX, int robotY) { // Fonction pour initialiser la grille
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            grille[i][j] = '.'; // Remplir la grille avec des espaces vides
        }
    }
    // Ajouter quelques obstacles
    grille[1][1] = 'X'; grille[3][5] = 'X';grille[6][7] = 'X';
    grille[robotX][robotY] = 'R'; // Placer le robot à la position donnée
}

void afficherGrille(char grille[ROWS][COLS]) {// Fonction pour afficher la grille
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("%c ", grille[i][j]);
        }
        printf("\n");
    }
}

void deplacerRobot(char grille[ROWS][COLS], int *robotX, int *robotY, int direction) {// Fonction pour déplacer le robot
    grille[*robotX][*robotY] = '.';// Retirer l'ancien emplacement du robot
    int newX = *robotX,newY = *robotY;// Calculer la nouvelle position selon la direction

    switch (direction) {
        case 'H': newX--; break; // Haut
        case 'B': newX++; break; // Bas
        case 'G': newY--; break; // Gauche
        case 'D': newY++; break; // Droite
    }

    // Vérifier si le mouvement est possible (pas d'obstacle, ni hors de la grille)
    if (newX >= 0 && newX < ROWS && newY >= 0 && newY < COLS && grille[newX][newY] != 'X') {
        *robotX = newX;// Mettre à jour la position du robot
        *robotY = newY;
    } else {
        printf("Mouvement impossible (obstacle ou hors de la grille).\n");
    }

    grille[*robotX][*robotY] = 'R';// Remettre le robot dans la nouvelle position
}
```

```c

int main() {
    char grille[ROWS][COLS];
    int robotX = ROWS / 2,robotY = COLS / 2; // Position initiale du robot au centre de la grille

    initGrille(grille, robotX, robotY); // Initialiser la grille avec le robot
    afficherGrille(grille); // Afficher la grille initiale

    char commande;// Boucle principale pour les commandes utilisateur
    while (1) {
        printf("\nEntrez une commande (H = haut, B = bas, G = gauche, D = droite, Q = quitter) : ");
        scanf(" %c", &commande); // Lire la commande utilisateur (avec un espace pour ignorer les espaces blancs)

        if (commande == 'H' || commande == 'B' || commande == 'G' || commande == 'D') {
            deplacerRobot(grille, &robotX, &robotY, commande);
        } else if (commande == 'Q') {
            break; // Quitter la boucle
        } else {
            printf("Commande non reconnue.\n");
        }
        afficherGrille(grille); // Afficher la grille après chaque action
    }
    return 0;
}
```