1 Matrices

Dans ces exemples on construit un tableau à 2 dimensions de taille 10×10 en utilisant des pointeurs sur pointeurs : `double **tableau` est un tableau de pointeurs, dont la taille est le nombre de ligne, et dont chaque élément (`*tableau`) est lui-même un pointeur sur un tableau de `double` dont la taille est le nombre de colonnes.

1.1 Allocation sans vérification

```
1  double **tableau;
2
3  tableau = (double **) malloc(sizeof(double *)*10);
4  for (i=0; i<10; i++)
5      tableau[i] = (double *) malloc(sizeof(double)*10);
```

1.2 Allocation avec vérification

```
1  double **tableau;
2
3  tableau = (double **) malloc(sizeof(double *)*10);
4  if (tableau != NULL)
5    {
6      i=0;
7      erreur=0;
8      while ((i<10) && !erreur)
9        {
10         tableau[i] = (double *) malloc(sizeof(double)*10);
11         if (tableau == NULL)
12             erreur = 1;
13         else
14             i++;
15       }
16     if (erreur)
17       {
18         while (i)
19           {
20             i--;
21             free(tableau[i]);
22           }
23       }
24   }
```

1.3 Initialisation

```
1  for (i=0; i<10; i++)
2      for (j=0; j<10; j++)
3          tableau[i][j] = 0;
```

2 Segmentation fault

Expliquez pourquoi le programme suivant génère une erreur de segmentation lors de son exécution.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  typedef struct {
6    char *nom;
7    char *prenom;
8    double moyenne;
9  } type_eleve;
10
11
12 void main() {
13   type_eleve *eleve;
14
15   eleve->nom = (char *) malloc(sizeof(char));
16   strcpy(eleve->nom, "Einstein");
17   strcpy(eleve->prenom, "Albert");
18   eleve->moyenne = 19.99;
19   printf("Nom : %s\n", eleve->nom);
20   printf("Prenom : %s\n", eleve->prenom);
21   printf("Moyenne : %s\n", eleve->moyenne);
22 }
```