Le code ci-dessous calcul la multiplication entre deux matrices carrées – dont la taille est passée en paramètre. Le calcul s'effectue sur le GPU à l'aide du framework Aparapi. Après avoir pris connaissance du code, répondez aux 3 questions ci-dessous en cochant les cases.
***Note** : il n'est pas nécessaire de comprendre l'algorithme utilisé, mais plutôt l'utilisation du GPU.*

1. Quel est votre niveau d'expérience en programmation ?

Très grande expérience en programmation	1	2	3	4	5	Aucune expérience en programmation

2. Quel est notre niveau d'expérience en programmation GPU ?

Très grande expérience en programmation GPU	1	2	3	4	5	Aucune expérience en programmation GPU

3. Comment jugez-vous la difficulté à comprendre l'utilisation du GPU dans le code ?

Très difficile à comprendre	1	2	3	4	5	Très facile à comprendre

```java
import java.util.Random;
import com.amd.aparapi.Kernel;

public class GPMatrix {

  public static int verbose = 0; // level 0,1,2
  public static int size;

  public static void main(String[] args) {

    // Params handling
    if(args.length < 1) {
      System.err.println("Needs size argument");
      System.exit(0);
    }
    size = Integer.valueOf(args[0]);

    // Matrix declaration
    double[][] A = new double[size][size];
    double[][] B = new double[size][size];
    double[][] C = new double[size][size];

    matrixInit(A,B);

    // --------------------------------
    long startTime = System.currentTimeMillis();
    // --- Start of benchmark zone --->
    new AparapiMatrixMul(A,B,C, size).execute(size);
    // <--- End of benchmark zone -----
    long stopTime = System.currentTimeMillis();
    // --------------------------------

    // Benchmark time elapsed computation
    long elapsedTime = stopTime - startTime;

    // Output
    if(verbose > 0) System.out.println("Time elapsed: " + elapsedTime/1000 + "s");
    System.out.println(elapsedTime/1000);

  }
```

```
41
42    // Matrix random initialization of A and B
43    public static void matrixInit(double[][] A, double[][] B) {
44      Random r = new Random();
45      for(int row=0; row < size; row++) {
46        for(int col=0; col < size; col++) {
47          A[row][col] = r.nextDouble();
48          B[row][col] = r.nextDouble();
49        }
50      }
51    }
52
53  }
54
55  // Is the class that sends job on the GPU
56  class AparapiMatrixMul extends Kernel {
57
58    double[][] A; // Matrix A
59    double[][] B; // Matrix B
60    double[][] C; // Matrix C, the result of AxB
61    int size;
62
63    public AparapiMatrixMul(double[][] A, double[][] B, double[][] C, int size) {
64      this.A = A; this.B = B; this.C = C;
65      this.size = size;
66    }
67
68    @Override
69    // Kernel method
70    public void run() {
71      int i = getGlobalId(); // The id of the thread
72
73      for(int j=0; j < size; j++) {
74        double sum = 0.0;
75        for(int k=0; k < size; k++) {
76          sum += A[i][k] * B[k][j];
77        }
78        C[i][j] = sum;
79      }
80    }
81
82  }
```