Examen

Exercice 1
```
public static double algo(double n){
  if(n<1.0) return 1.0;
  int x=algo(n/2)+algo(n/2)+algo(n/2)+algo(n/2);
  for(int i=0; i<n; i++) x+=i+1;
  return x;
}
```

1) Que renvoient `algo(0)`, `algo(1)`, `algo(2)`, `algo(4)`, `algo(8)` ?

2) Quelle est la complexité de `algo(n)` ?

3) Modifier `algo(n)` pour que la complexité soit $\Theta(n)$.

4) Modifier `algo(n)` pour que la complexité soit $\Theta(\lg n)$.

Exercice 2
```
public class Queens {
  final int SIZE=8;
  int [] solution;
  boolean [] ligne, diag1, diag2;
  public Queens(){
    solution = new int[SIZE];
    ligne = new boolean[SIZE];
    diag1 = new boolean[2*SIZE-1];
    diag2 = new boolean[2*SIZE-1];
    for (int j=0; j < SIZE; ++j)
      ligne[j]=diag1[j]=diag2[j]=true;
    for (int j=SIZE; j < 2*SIZE-1; ++j)
      diag1[j]=diag2[j]=true;
  }
  void essayer(int c){
    for (int l=0; l < SIZE; ++l){
      if(ligne[l] && diag1[l+c] && diag2[l-c+SIZE-1]){
        solution[c]=l;
        ligne[l]=diag1[l+c]=diag2[l-c+SIZE-1]=false;
        if(c!=SIZE-1) essayer(c+1);
        ligne[l]=diag1[l+c]=diag2[l-c+SIZE-1]=true;
      }
    }
  }
}
```

1) Après la déclaration `Queens Q = new Queens()` et lors de l'appel `Q.essayer(0)`, donnez le contenu du tableau `solution` juste après que le test

`if(ligne[l] && diag1[l+c] && diag2[l-c+SIZE-1])`

soit faux pour la première fois.

2) Après que le test de la question 1) soit redevenu vrai pour la première fois donnez le contenu du tableau `solution` juste après l'exécution de l'instruction `solution[c]=l`.

Exercice 3 Que fait l'algorithme de Strassen et quelle est sa complexité ?

Que fait l'algorithme de Karatsuba et quelle est sa complexité ?

Exercice 4 On obtient une approximation y de la racine carrée $\sqrt{x}$ de x en initialisant $y := x$ puis en itérant le procéssus $y := \frac{1}{2}(y + (x/y))$ un certain nombre n de fois ; plus on itère ce procéssus, meilleure est l'approximation. Programmez les fonctions suivantes en Java:

1) `double myAbs(double x)` renvoyant la valeur absolue $|x|$ de x.

2) `double mySqrt(double x, int n)` retournant la valeur de l'approximation de $\sqrt{x}$ lorsqu'on itère n fois le procéssus.

3) `boolean IsLess(double x, int n, double approx)` testant si l'erreur relative $\frac{|y-\sqrt{x}|}{\sqrt{x}}$ après n itérations est inférieure à `approx`.

4) `int ForAccuracy(double x, double approx)` renvoyant le nombre minimum d'itérations minimum pour obtenir la précision `approx`.

Exercice 5 Donnez le code d'une fonction Java `double pow(double x, int n)` qui éleve x à la puissance n en $O(\lg n)$.

Exercice 6 1) Permuter les éléments du tableau $A = (5,7,8,14,45,2,71,8,55,13,46)$ de manière à en faire un tas min.

2) Elaborer un algorithme de tri en $O(n \lg n)$ en utilisant un tas min.