**Ejercicio 1:** Determine la complejidad de cada método utilizando la notación ***O grande***. Justifique su respuesta empleando las fórmulas y realizando los cálculos correspondientes.

|  |  |
| --- | --- |
| **void** Algorit(**int**[] vector, **int** n) {  **int** a = 0;  **if** (n % 2 == 0) {  a = Algorit1(vector, n - vector[n]);  } **else** {  **int**[] v1 = new **int**[]{2, 7, 5, 8, 3, 4,};  a = Algorit1(vector, n) \* Algorit2(v1, v1.lenght/2);  }  System.out.println(a);  } | |
| **int** Algorit1(**int**[] vector, **int** n) {  **int** var = 0;  **for** (**int** j = 0; j < n; j++) {  **int** i = n - 1;  **while** (i >= 0) {  **if** (vector[i] % 2 == 0) {  var += vector[i];  }  i = i / 2;  }  }  **return** var;  } | **int** Algorit2(**int**[] vector, **int** n) {  **int** var = 0, i = 0;  **while** (n < 0) {  **if** (i % 2 == 0) {  var += vector[i];  }  i++;  }  **return** var;  } |

**Ejercicio 2:** Dado el siguiente diagrama de clases en UML:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | NodoDE<T> | | # info:T  # siguiente:NodoDE<T>  # anterior:NodoDE<T> | | + getInfo():T  + setInfo(info:T)  + getSiguiente():NodoDE<T>  + setSiguiente(sig:NodoDE<T>)  + getAnterior():NodoDE<T>  + setAnterior(ant:NodoDE<T>) | | |  | | --- | | ListaDE<T> | | - cabeza:NodoDE<T> | | + EliminarCentro():void |   **Por ejemplo**:  (2, **6, 3**, 7) 🡪EliminarCentro🡪 (2, 7)  (3,7,**1**,4,6) 🡪EliminarCentro🡪 (3,7,4,6) |

1. Se desea implementar una funcionalidad de modo que al eliminar elimine siempre el elemento que se encuentra al centro de la lista si la cantidad de elementos que contiene es impar y en caso contrario elimine los dos elementos centrales.

**Ejercicio 3:** Para simular el funcionamiento de un hospital se han definido las siguientes clases:

|  |
| --- |
| **Hospital** |
| - pacientes : Queue<Paciente>  - embarazadas : Queue<Paciente> |
| **+ AtenderPaciente(cant : int) : LinkedList<Paciente>**  … |

|  |
| --- |
| **Paciente** |
| -nombre : String  -edad : int  -sintomas : LinkedList<String> |
| + getNombre() : String  + getEdad() : int  + getEnfermedad() : Padecimiento()  … |

Implemente el método **AtenderPacientes(cant : int) : LinkedList<Paciente>** que dado una cantidad de pacientes devuelve una lista con los atendidos, teniendo en cuenta que por cada dos embarazadas se atiende un paciente de la otra cola. Las colas originales de pacientes deben mantenerse solamente con aquellos que aún no han sido atendidos.