Control 1: Algoritmos Aritméticos y Análisis Big O
Profesor Yerko Ortiz
Tiempo: 50 minutos Nombre: _____

Verdadero o Falso

Responda V para las verdaderas y F para las falsas. Justifique las falsas. 4 puntos cada respuesta correcta. Respuestas falsas sin justificar se considerarán incorrectas.

1. __F__ Un algoritmo es una secuencia de pasos bien definida que puede tener duración infinita.

   ```
   Un algoritmo debe tener duración finita.
   ```

2. __F__ Mientras mejor CPU y memoria tenga un computador, mejor será la complejidad Big O de un algoritmo.

   ```
   La complejidad Big O describe el peor caso de forma
   independiente a la CPU o memoria específica que tenga
   una maquina o dispositivo
   ```

3. __V__ Una estructura de datos ofrece un conjunto de operaciones para administrar los datos que almacena.

4. __F__ La notación Big O describe el mejor caso de un algoritmo.

   ```
   Describe el peor caso
   ```

5. __F__ El sistema decimal (base 10) utiliza 9 dígitos para representar valores numéricos.

   ```
   Falso usa 10 dígitos {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
   ```

6. __F__ El siguiente algoritmo tiene un tiempo de ejecución $\mathcal{O}(N)$

```java
static void f1(int N){
    while(N > 0) {
        // <body>
        N/=2;
    }
}
```

   ```
   Es O(log N), en cada iteración N decrece a la mitad
   ```

7. __F__ El siguiente algoritmo tiene un tiempo de ejecución $\mathcal{O}(N + M)$

```java
static void f2(int N, int M){
    for(int i = 0; i < N; ++i) {
        for(int j = 0; j < M; ++j) {
            // <body>
        }
    }
}
```

   ```
   Es O(N * M), se hacen M operaciones N veces
   ```

8. __F__ El siguiente algoritmo tiene un tiempo de ejecución $\mathcal{O}(\log N)$

```
static void f3(){
    for(int i = 0; i < 1024; i*=2) {
        // <body>
    }
}
```

Es O(1), es constante.

9. __F__ Sea el siguiente algoritmo que recibe como entrada un arreglo de dígitos entre 0 y 7. La complejidad Big O de dicho algoritmo es $\mathcal{O}(\log N)$

```
static int misteriousAlgorithm(int[] digits) {
    int result = 0;
    for (int i = 0; i < digits.length; i++) {
        result = result * 8 + digits[i];
    }
    return result;
}
```

Es O(N), depende linealmente del largo del arreglo

10. __F__ Sea el siguiente algoritmo que recibe como entrada un arreglo de dígitos entre 0 y 7. Al evaluar el algoritmo para el arreglo $\{7, 7, 7\}$ el resultado del algoritmo es 500.

```
static int misteriousAlgorithm(int[] digits) {
    int result = 0;
    for (int i = 0; i < digits.length; i++) {
        result = result * 8 + digits[i];
    }
    return result;
}
```

Es 511

Intertir un entero

En el país invergonia les gusta invertir cosas, en esta ocasión están estudiando cómo invertir los dígitos de un número entero N, donde por ejemplo si N=135, el resultado debería ser 531. Usted como invitado especial a este país, tendrá el honor de diseñar un algoritmo que reciba como entrada un número entero (int) N y retorne otro número entero como resultado, dicho resultado es un número M con los dígitos de N invertidos. Diseñe un algoritmo en Java o Pseudocódigo, que sea capáz de invertir cualquier número entero positivo N. Analice su solución utilizando la notación Big O.
Input:

- Un número entero N.

Output:

- Un número entero M, dicho entero tiene los dígitos de N pero en orden inverso.

Ejemplo 1: Si N = 425, entonces M=524

Ejemplo 2: Si N = 1024, entonces M=4201

Hint: El input es un número entero (int), no un arreglo. Asuma que N será siempre positivo.

1. La solución propuesta en Java o Pseudocódigo, tiene un puntaje de 15 puntos.
2. Analizar la solución propuesta en notación Big O, tiene un puntaje de 5 puntos.

`Este problema tenía múltiples soluciones considerando como input un entero N.`

La soluciones que tienen el puntaje total en particular eran las siguientes:

1. Calcular el resultado In Place, este método tiene una complejidad de $\mathcal{O}(d)$, donde d es la cantidad de dígitos en el entero, o también se considera correcto si se analiza respecto el valor de N, donde la complejidad sería $\mathcal{O}(lgN)$.

```java
static int invertInteger1(int N) {
    int result = 0;
    while (N != 0) {
        result = result * 10 + (N % 10);
        N = N / 10;
    }
    return result;
}
```

2. Calcular el resultado utilizando un arreglo auxiliar. Este método tiene una complejidad de $\mathcal{O}(d)$, donde d es la cantidad de dígitos en el entero, o también se considera correcto si se analiza respecto el valor de N, donde la complejidad sería $\mathcal{O}(lgN)$.

```java
static int invertInteger2(int N) {
    int result = 0;
    int auxArray[] = new int[(int) Math.log10(N) + 1];
    int i = 0;
    while (N != 0) {
        auxArray[i] = N % 10;
        N = N / 10;
        i++;
    }

    for (int j = 0; j < auxArray.length; j++) {
        result = result * 10 + auxArray[j];
    }
    return result;
}
```

3. Calcular el resultado de forma recursiva. Este método tiene una complejidad de $\mathcal{O}(d)$, donde d es la cantidad de dígitos en el entero, o también se considera correcto si se analiza respecto el valor de N, donde la complejidad sería $\mathcal{O}(lgN)$.

```java
static int invertInteger3(int N) {
    if (N == 0) {
        return 0;
    }
    return (
        (N % 10) * (int) Math.pow(10, (int) Math.log10(N)) +
        invertInteger3(N / 10)
    );
}
```

Para evaluar este problema se utilizarán los siguientes criterios a modo de rúbrica:

1. 3 Puntos por claridad y orden en la solución, si la solución entregada ya sea en pseudocódigo, Java o C++ es descrita instrucción a instrucción entonces cae en este criterio.
2. 5 Puntos por hacer uso del sistema posicional (N%10) y (N/10) para obtener los dígitos de manera individual.
3. 7 Puntos por la correctitud del algoritmo diseñado. Acá el algoritmo es correcto o no lo es.
4. 5 Puntos por analizar la solución propuesta de forma correcta.