**Ejecutable G**

**.Lista de casos en los que falla:**

**06\_ConsiderandoCerosEnAmbos:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 6  001256  4  0125 | SI 1  2 |

|  |
| --- |
| Salida del ejecutable |
| NO |

*Observaciones:* Evidentemente el programa nunca llega a ver el número escondido de forma completa.

**09\_ElNúmeroContenedorEstáCompletamenteFormadoPorAparicionesDelNúmeroEscondido:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 12  467467467467  3  467 | SI 10  1 2 3 4 5 6 7 8 9 10 |

|  |
| --- |
| Salida del ejecutable |
| SI 4  1 4 7 10 |

*Observaciones:* El ejecutable se saltea valores internos, pero no se puede afirmar que sea incapaz de reconocer valores desordenados o superpuestos.

**10\_ElSegundoEstáIncluidoDeAmbasFormasUnaVez:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 9  486816841  3  486 | SI 2  1 6 |

|  |
| --- |
| Salida del ejecutable |
| SI 1  1 |

*Observaciones:* El programa es incapaz de reconocer al número a buscar si este está desordenado. Sin embargo, puede ocurrir que este simplemente esté fuera de su rango de búsqueda, es decir, que nunca llegue a “verlo”.

**11\_ElSegundoEstáIncluidoDeAmbasFormasVariasVeces:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 18  486816841486816841  3  486 | SI 4  1 6 10 15 |

|  |
| --- |
| Salida del ejecutable |
| SI 2  1 10 |

*Observaciones:* Ocurre algo muy similar al caso anterior, solo que esta vez el valor está incluido una mayor cantidad de veces.

**12\_ElSegundoEstáIncluidoExactamenteIgualUnaVez:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 8  48689895  3  895 | SI 1  6 |

|  |
| --- |
| Salida del ejecutable |
| NO |

*Observaciones:* En este caso, incluso si el número a buscar se encuentra exactamente de la misma forma en el número contenedor, el programa no es capaz de encontrarlo. Esto nos lleva a pensar que en efecto, el problema no tiene que ver con que los números estén escritos de forma desordenada.

**13\_ElSegundoEstáIncluidoExactamenteIgualVariasVeces:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 9  156897568  3  568 | SI 2  2 7 |

|  |
| --- |
| Salida del ejecutable |
| SI 1  7 |

*Observaciones:* Similar al caso anterior, solo que esta vez el número está incluido varias veces, nuevamente, escrito de la misma manera.

**14\_ElSegundoEstáIncluidoNoExactamenteIgualUnaVez:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 7  1536985  3  936 | SI 1  3 |

|  |
| --- |
| Salida del ejecutable |
| NO |

*Observaciones:* En este caso el programa no encuentra el número. Se puede destacar que el número tiene los dígitos desordenados, pero en base a observaciones anteriores esto no parece relevante.

**15\_ElSegundoEstáIncluidoNoExactamenteIgualVariasVeces:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 10  1256486257  3  652 | SI 2  2 7 |

|  |
| --- |
| Salida del ejecutable |
| SI 1  7 |

*Observaciones:* Finalmente el programa es capaz de encontrar un número desordenado, descartando casi por completo la incapacidad de este de identificarlos. Se considera que el problema está relacionado a un rango de búsqueda en específico.

**20\_TodasLasAparicionesDelEscondidoSeSuperponen:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 14  84747474747569  2  74 | SI 9  2 3 4 5 6 7 8 9 10 |

|  |
| --- |
| Salida del ejecutable |
| SI 4  3 5 7 9 |

*Observaciones:* Nuevamente el programa se saltea los valores superpuestos. Como ya sabemos que no tiene imposibilitado reconocer valores escritos de forma desordenada, se asume que está definiendo un rango de búsqueda con la información que tiene disponible en la entrada, lo cual está provocando que el programa obvie determinados números que son válidos.

**21\_TodosLosDígitosDeAmbosSonElMismoDígito:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 6  777777  3  777 | SI 4  1 2 3 4 |

|  |
| --- |
| *Salida del ejecutable* |
| SI 2  1 4 |

*Observaciones:* El problema que tiene el programa para identificar valores superpuestos persiste. Su rango de búsqueda evidentemente hace que se saltee dichos valores. Curiosamente este rango puede estar dado por el número de dígitos **M (cantidad de dígitos del número escondido)**. Esta deducción verifica para todos los casos anteriores.

\*Con las observaciones recopiladas hasta el momento, se plantean casos de prueba adicionales que prueben que las deducciones obtenidas hasta ahora sean correctas. Últimamente, se procederá a dar una conclusión final en cuanto al funcionamiento del programa. \*

**CASOS ADICIONALES:**

|  |  |
| --- | --- |
| Entrada | Salida |
| 15  467146714671467  3  467 | SI 4  1 5 9 13 |

|  |
| --- |
| Salida del ejecutable |
| SI 2  1 13 |

|  |  |
| --- | --- |
| Entrada | Salida |
| 16  14671467146711467  3  467 | SI 4  2 6 10 15 |

|  |
| --- |
| Salida del ejecutable |
| SI 1  10 |

|  |  |
| --- | --- |
| Entrada | Salida |
| 12  764764764764  3  467 | SI 10  1 2 3 4 5 6 7 8 9 10 |

|  |
| --- |
| Salida del ejecutable |
| SI 4  1 4 7 10 |

**.Conclusiones:**

Lo que está haciendo el ejecutable G es tomar la cantidad de dígitos **M** que tiene el valor escondido, y usar esa cantidad para recorrer el número con cantidad de dígitos **N**. En otras palabras, lo que hace es “partir” el número con **N** dígitos en partes de **M** dígitos y valida cada una de esas partes teniendo en cuenta que el número escondido puede tener los dígitos desordenados. Esto provoca que se saltee valores internos que son válidos, lo cual queda especialmente claro cuando se da un caso como **467467467** **(N=9)** siendo el número a buscar **467 (M=3)** ya nunca llega a tener en cuenta el valor **674 (el cual es válido)** debido a que no es un número que forme parte del recorrido de validación que el programa realiza, lo cual es erróneo.