Lote de Prueba “Lanzando el cáber”

# Caso00\_Enunciado

Se busca encontrar el error al reproducir el caso ejemplo del enunciado.

|  |  |
| --- | --- |
| .in | .out |
| 3  1.49 5.00  1.50 3.00  1.51 -3.00  2.10 6.00  2.00 -6.00  1.00 2.00  2.30 -35.00  1.30 -92.00  1.00 1.00 | 1 2  2 1 3 |

# Caso01\_OrdenamientoPodioDistancia

Busca encontrar error al ordenar el podio según la distancia total de cada participante.

|  |  |
| --- | --- |
| .in | .out |
| 3  3.23 2.08  2.52 1.05  4.55 4.51  5.64 4.90  10.8 3.74  5.28 3.26  10.7 4.79  10.5 4.21  10.2 1.05 | 3 2 1 |

# Caso02\_TiroDescalificado

Busca encontrar error al descalificar un tiro por tener un ángulo invalido.

|  |  |
| --- | --- |
| .in | .out |
| 4  4.83 1.30  3.96 4.91  0.68 0.29  4.60 0.45  2.16 2.72  1.78 1.23  0.04 1.20  3.37 4.80  4.29 4.46  4.96 0.67  3.46 0.22  3.95 -100.15 | 2 3 1  1 2 4 |

# Caso03\_OrdenamientoPodioConsistencia

Busca encontrar error al ordenar el podio según la consistencia de cada participante.

Dado que no se puede asegurar que el error sea exclusivamente de este podio, si el test 1 no falla, significa que el error se encuentra en este caso.

|  |  |
| --- | --- |
| .in | .out |
| 5  1.34 3.42  2.75 2.48  3.63 0.40  3.91 0.06  1.33 2.01  2.64 1.66  1.88 3.34  0.50 3.25  1.04 0.23  4.76 2.58  0.77 2.76  3.86 1.61  2.70 3.94  3.19 4.70  2.45 1.94 | 5 2 3  4 5 2 |

# Caso04\_FlujoNormal

Se busca encontrar el error al probar el flujo normal del programa con 6 lanzadores.

|  |  |
| --- | --- |
| .in | .out |
| 6  3.60 0.83  1.71 2.55  2.44 4.16  3.66 4.14  2.33 3.97  1.89 0.94  3.11 1.29  4.27 1.29  1.10 1.69  0.73 1.57  2.27 0.25  2.88 2.52  2.77 1.99  3.90 3.95  0.44 3.37  3.84 0.43  0.69 2.55  3.77 4.91 | 3 4 1  3 6 2 |

# Caso05\_PodioConsistenciaIncompleto

Busca encontrar error al tener el podio de consistencia incompleto.

|  |  |
| --- | --- |
| .in | .out |
| 3  10.00 20.00  10.00 20.00  10.00 20.00  200.00 30.00  200.00 20.00  200.00 10.00  100.00 100.00  10.00 20.00  10.00 20.00 | 1 2  2 1 3 |

# Caso06\_SinPodioConsistencia

Busca encontrar error al descalificar los participantes que tengan tiros con ángulos inválidos en el podio de consistencia.

|  |  |
| --- | --- |
| .in | .out |
| 3  10.00 20.00  10.00 20.00  10.00 100.00  200.00 100.00  200.00 100.00  200.00 100.00  100.00 100.00  10.00 20.00  10.00 20.00 | 1 3 |

# Caso07\_DistanciaAl80

Busca encontrar error al corregir la distancia del tiro del participante cuando su ángulo esta entre las 9 y las 11, o entre la 1 y las 3.

|  |  |
| --- | --- |
| .in | .out |
| 3  100.00 20.00  100.00 20.00  130.00 20.00  100.00 20.00  100.00 20.00  150.00 80.00  100.00 10.00  100.00 20.00  150.00 25.00 | 1 3 2  3 1 2 |

# Caso08\_SinPodios

Busca encontrar error al descalificar todos los tiros de todos los participantes por tener ángulos inválidos.

|  |  |
| --- | --- |
| .in | .out |
| 3  10.00 100.00  10.00 100.00  10.00 100.00  200.00 100.00  200.00 100.00  200.00 100.00  100.00 100.00  10.00 100.00  10.00 100.00 |  |

# Caso09\_CasoFatiga

Para el caso de fatiga generamos un caso con 1.000.000 de participantes donde, cada participante tenga mayor consistencia y la menor distancia que los siguientes, y menor consistencia y mayor distancia que los anteriores.

Por lo cual, en cada iteración del programa, va a intercambiar participantes en los podios, y la salida esperada van a ser los últimos 3 participantes para ambos podios.

|  |  |
| --- | --- |
| .in | .out |
| 1000000  1000000.00 0.00  2000000.00 0.00  3000000.00 0.00  3999999.00 0.00  4999998.00 0.00  5999997.00 0.00  6999995.00 0.00  7999993.00 0.00  8999991.00 0.00  9999988.00 0.00  10999985.00 0.00  11999982.00 0.00  12999978.00 0.00  … | 1000000 999999 999998  1000000 999999 999998 |