
Centro de masas de partículas que se mueven a velocidad constante X30485_es

Preliminares

En este ejercicio implementaremos un programa que escribe números reales por la salida. Debido a problemas de formato y eficiencia de `cin` y `cout` cuando trabajan con números reales, conviene hacer algunas cosas que mencionamos a continuación.

Al principio de la función `main` debéis poner estas instrucciones:

```
ios::sync_with_stdio(false);
cin.tie(0);
cout.setf(ios::fixed);
cout.precision(5);
```

Para escribir reales por la salida, conviene que incluyáis y uséis esta función:

```
void printDouble(double d)
{
    if (abs(d) < 1e-7)
        cout << 0.0;
    else
        cout << d;
}
```

Por último, y no por ello menos importante, no uséis `endl` para escribir saltos de línea. Usad `'\n'` en su lugar. Es decir:

```
// This line has been replaced with next one:
// cout << endl;
cout << '\n';
```

Ejercicio

Nota: En este ejercicio hablaremos de posiciones de partículas en un sistema 3-dimensional, y de sus velocidades y masas. El sistema de referencia y unidades no es demasiado relevante, siempre y cuando se escoja uno de más o menos estandar y razonable. Por si alguien siente la necesidad, os podéis imaginar que hablamos de metros, segundos, metros por segundo y kilogramos.

Tenemos como entrada la posición, velocidad y masa de n partículas $(p_1, v_1, m_1), \dots, (p_n, v_n, m_n)$. En particular, la primera partícula se encuentra en posición p_1 en tiempo 0, se desplaza a velocidad constante v_1 , y tiene masa m_1 . Se supone que todo se expresa en un sistema cartesiano de coordenadas en tres dimensiones.

También tenemos como entrada k valores de tiempo t_1, \dots, t_k .

Tendremos que determinar el centro de masas de las n partículas después de t_1 unidades de tiempo, después de $t_1 + t_2$ unidades de tiempo, después de $t_1 + t_2 + t_3, \dots$, después de $t_1 + \dots + t_k$ unidades de tiempo.

Es obligatorio usar coherentemente las siguientes declaraciones de tipos, y implementar y utilizar coherentemente las siguientes funciones. En caso contrario, se invalidará la entrega. Podéis declarar más tipos de datos y implementar más funciones si queréis, y de hecho es recomendable hacerlo.

```

struct Point {
double x, y, z;
};

struct Particle {
Point p,v;
double m;
};

// Pre:
// Post: returns the sum of p1 and p2.
Point sum(Point p1, Point p2)
{
//...
}

// Pre:
// Post: returns a times p.
Point mul(double a, Point p)
{
//...
}

```

Nota: Os recomendamos empezar haciendo una implementación sencilla con el fin de superar los juegos de pruebas públicos, y tratar después de optimizarla con el fin de superar también los juegos de pruebas privados.

Entrada

La entrada tiene varios casos. Cada caso comienza con dos naturales positivos n, k en una primera línea. Después vienen n líneas, cada una describiendo la posición (tres enteros), velocidad (tres enteros) y masa (un natural positivo) de una partícula. Finalmente vienen k líneas, cada una con un natural positivo que representa un tiempo transcurrido.

Salida

Para cada caso, primero se han de escribir k líneas, donde la i -ésima línea continene el centro de masas (tres reales redondeados a 5 dígitos después del punto decimal) de todas las partículas después de la suma de los i primeros tiempos. Finalmente, se han de escribir n líneas, con las posiciones (tres reales redondeados a 5 dígitos después del punto decimal) de las partículas después de la suma de todos los instantes de tiempo. Cada caso viene seguido de una línea en blanco.

Ejemplo de entrada 1

```

2 2
0 0 0 0 0 1 1
1 0 0 0 0 1 1
1
1
2 2
0 0 0 1 0 0 1

```

```

1 0 0 1 0 0 1
1
1
2 2
0 0 0 1 0 0 1
1 0 0 0 0 1 1
1
1

```

Ejemplo de salida 1

0.50000 0.00000 1.00000
0.50000 0.00000 2.00000
0.00000 0.00000 2.00000
1.00000 0.00000 2.00000

1.50000 0.00000 0.00000
2.50000 0.00000 0.00000

Ejemplo de entrada 2

4 2
1 -3 -4 -1 -5 1 5
-4 3 2 0 -2 2 2
4 5 -3 -5 5 3 3
-5 -1 1 -5 5 -2 1
5
3
3 4
2 4 2 3 -3 5 5
5 -1 -4 -5 5 -1 5
2 1 3 1 -3 -3 2
4
3
1
2
2 1
-1 -1 -2 1 4 5 5
-1 -4 -5 5 4 -1 5
3
5 5
1 4 2 -4 -1 4 4
4 -3 5 -2 2 1 1
4 3 4 -1 0 4 5
4 3 -2 3 2 0 2
-1 2 2 3 0 -4 2
1
2
2
1
5
5 5
-1 -3 -1 -2 5 -2 2
-3 5 3 5 0 -5 4
5 1 5 -2 -5 2 4
0 1 4 5 -1 4 4
-2 2 -3 3 2 -1 4
1
2
2
2
1
5 4
3 3 -5 2 -2 4 5
-4 0 3 -1 -5 1 2
-3 -3 4 1 2 4 3
-2 -2 3 3 3 4 1
-5 3 3 3 -1 1 4
2
4

2.00000 0.00000 0.00000
3.00000 0.00000 0.00000

1.00000 0.00000 0.50000
1.50000 0.00000 1.00000
2.00000 0.00000 0.00000
1.00000 0.00000 2.00000

5
2
5 4
-3 1 -2 -1 3 1 4
-4 -5 5 -5 -1 -5 5
5 2 2 -4 2 -3 1
-3 1 -3 -1 -4 0 5
3 5 -1 3 -2 1 5
4
2
3
4
5 4
-4 -4 -4 2 -5 3 1
-4 -3 -1 3 -1 5 2
3 4 -2 -3 1 -5 3
-3 -1 -5 -1 -2 3 2
-2 2 0 -3 4 -1 4
2
2
1
4
5 1
-1 0 -1 -3 -4 -5 4
-4 2 -1 1 4 1 5
-5 4 -4 0 -4 2 3
-3 -2 -3 -4 5 -4 4
1 -4 1 -5 -1 3 3
2
2 4
0 -3 -1 4 4 5 5
4 3 -3 -4 2 2 3
1
5
4
1
5 3
5 -2 3 -2 -4 -3 3
-3 0 1 -1 5 2 1
4 0 0 5 -3 -3 1
-4 2 2 4 1 -1 5
-4 -4 -4 -5 0 4 1
1
3
5
5 4
1 3 -2 -3 -2 5 5
3 4 2 3 2 2 5
-2 0 4 0 5 5 4
-5 4 5 4 5 -2 4
5 5 -4 -2 2 -3 2

4
 5
 1
 2
 4 3
 -1 2 5 4 3 -3 4
 -2 -4 -1 3 5 -5 4
 2 -5 5 -1 3 2 1
 -5 -3 0 -2 -5 2 3
 5
 3
 5
 4 5
 -5 1 4 -2 -5 5 4
 -2 0 -3 -2 -4 4 1
 4 -5 4 1 -2 2 5
 1 5 1 -2 4 -4 5
 5
 3
 4
 3
 1
 4 5
 -1 -3 -4 0 2 -4 2
 1 5 3 0 3 -1 2
 -5 -4 -1 2 4 5 3
 2 4 -1 0 3 -5 4
 1
 4
 3
 2
 1
 2 2
 5 -4 -2 -2 4 4 1
 2 -2 -1 0 -3 1 3
 3
 3
 2 5
 -5 4 4 3 3 3 5
 5 1 5 -4 0 0 3
 4
 4
 2
 1
 5
 2 3
 1 -5 -5 3 -1 4 3
 3 -2 -5 3 -3 4 3
 5
 4
 3
 4 3
 3 1 -1 -4 0 4 4
 2 5 -5 -3 0 2 4
 -2 -3 2 -4 2 -3 4
 2 3 0 -2 -5 -3 3
 1
 3
 2
 3 1
 -5 1 0 2 -4 5 2

-4 -5 2 1 -3 5 1
 2 -4 -2 2 -1 -5 2
 5

Ejemplo de salida 2

-11.00000 -3.63636 5.09091
-17.81818 -6.09091 9.45455
-7.00000 -43.00000 4.00000
-4.00000 -13.00000 18.00000
-36.00000 45.00000 21.00000
-45.00000 39.00000 -15.00000

0.58333 2.75000 4.33333
-1.41667 3.75000 7.83333
-2.08333 4.08333 9.00000
-3.41667 4.75000 11.33333
32.00000 -26.00000 52.00000
-45.00000 49.00000 -14.00000
12.00000 -29.00000 -27.00000

8.00000 9.50000 2.50000
2.00000 11.00000 13.00000
14.00000 8.00000 -8.00000

1.64286 2.85714 4.42857
0.07143 3.14286 8.57143
-1.50000 3.42857 12.71429
-2.28571 3.57143 14.78571
-6.21429 4.28571 25.14286
-43.00000 -7.00000 46.00000
-18.00000 19.00000 16.00000
-7.00000 3.00000 48.00000
37.00000 25.00000 -2.00000
32.00000 2.00000 -42.00000

2.11111 1.33333 1.66667
6.55556 0.66667 1.22222
11.00000 0.00000 0.77778
15.44444 -0.66667 0.33333
17.66667 -1.00000 0.11111
-17.00000 37.00000 -17.00000
37.00000 5.00000 -37.00000
-11.00000 -39.00000 21.00000
40.00000 -7.00000 36.00000
22.00000 18.00000 -11.00000

1.86667 -0.93333 6.13333
8.80000 -4.93333 17.33333
17.46667 -9.93333 31.33333
20.93333 -11.93333 36.93333
29.00000 -23.00000 47.00000
-17.00000 -65.00000 16.00000
10.00000 23.00000 56.00000
37.00000 37.00000 55.00000
34.00000 -10.00000 16.00000

-5.95000 -3.65000 -3.85000
-8.25000 -5.75000 -5.75000
-11.70000 -8.90000 -8.60000
-16.30000 -13.10000 -12.40000
-16.00000 40.00000 11.00000
-69.00000 -18.00000 -60.00000
-47.00000 28.00000 -37.00000
-16.00000 -51.00000 -3.00000

42.00000 -21.00000 12.00000

-3.91667 2.00000 -1.83333
-6.41667 3.33333 -1.83333
-7.66667 4.00000 -1.83333
-12.66667 6.66667 -1.83333
14.00000 -49.00000 23.00000
23.00000 -12.00000 44.00000
-24.00000 13.00000 -47.00000
-12.00000 -19.00000 22.00000
-29.00000 38.00000 -9.00000

-6.52632 1.05263 -3.26316
-7.00000 -8.00000 -11.00000
-2.00000 10.00000 1.00000
-5.00000 -4.00000 0.00000
-11.00000 8.00000 -11.00000
-9.00000 -6.00000 7.00000

2.50000 2.50000 2.12500
7.50000 18.75000 21.50000
11.50000 31.75000 37.00000
12.50000 35.00000 40.87500
44.00000 41.00000 54.00000
-40.00000 25.00000 19.00000

0.45455 -0.45455 0.45455
4.00000 -1.81818 -2.54545
9.90909 -4.09091 -7.54545
-13.00000 -38.00000 -24.00000
-12.00000 45.00000 19.00000
49.00000 -27.00000 -27.00000
32.00000 11.00000 -7.00000
-49.00000 -4.00000 32.00000

2.50000 11.85000 9.60000
5.50000 22.85000 19.85000
6.10000 25.05000 21.90000
7.30000 29.45000 26.00000
-35.00000 -21.00000 58.00000
39.00000 28.00000 26.00000
-2.00000 60.00000 64.00000
43.00000 64.00000 -19.00000
-19.00000 29.00000 -40.00000

6.66667 6.50000 -8.25000
11.91667 11.50000 -14.25000
20.66667 19.83333 -24.25000
51.00000 41.00000 -34.00000
37.00000 61.00000 -66.00000
-11.00000 34.00000 31.00000
-31.00000 -68.00000 26.00000

-4.80000 -4.40000 7.20000
-7.80000 -7.20000 10.00000
-11.80000 -10.93333 13.73333
-14.80000 -13.73333 16.53333
-15.80000 -14.66667 17.46667
-37.00000 -79.00000 84.00000
-34.00000 -64.00000 61.00000
20.00000 -37.00000 36.00000

-31.00000 69.00000 -63.00000	43.00000 52.00000 52.00000
	-59.00000 1.00000 5.00000
-0.09091 3.81818 -2.18182	
2.09091 16.18182 -7.63636	17.00000 -13.50000 15.00000
3.72727 25.45455 -11.72727	29.00000 -21.50000 31.00000
4.81818 31.63636 -14.45455	38.00000 -27.50000 43.00000
5.36364 34.72727 -15.81818	37.00000 -17.00000 43.00000
-1.00000 19.00000 -48.00000	39.00000 -38.00000 43.00000
1.00000 38.00000 -8.00000	
17.00000 40.00000 54.00000	-2.13333 0.93333 -0.86667
2.00000 37.00000 -56.00000	-12.13333 -0.46667 -0.26667
	-18.80000 -1.40000 0.13333
1.25000 -6.25000 4.00000	-21.00000 1.00000 23.00000
-0.25000 -10.00000 9.25000	-16.00000 5.00000 7.00000
-7.00000 20.00000 22.00000	-26.00000 9.00000 -16.00000
2.00000 -20.00000 5.00000	-10.00000 -27.00000 -18.00000
0.25000 10.37500 11.87500	7.00000 -15.20000 4.60000
1.75000 17.87500 19.37500	5.00000 -19.00000 25.00000
2.50000 21.62500 23.12500	1.00000 -20.00000 27.00000
2.87500 23.50000 25.00000	12.00000 -9.00000 -27.00000
4.75000 32.87500 34.37500	

Observación

Evaluación sobre 10 puntos:

- Solución lenta: 5 puntos.
- Solución rápida: 10 puntos.

Entendemos como solución rápida una que es correcta, de coste lineal y capaz de superar los juegos de pruebas públicos y privados. Entendemos como solución lenta una que no es rápida, pero es correcta y capaz de superar los juegos de pruebas públicos.

Información del problema

Autoría: PRO1

Generación: 2026-01-25T15:02:31.437Z

© *Jutge.org*, 2006–2026.

<https://jutge.org>