OMIPS 2022 - Solución para *Karel y los Chocolates*

Entiende el problema

¿Qué te piden hacer? Dados los números A, B y C, Karel debe determinar el cociente de $\frac{A+B}{C}$. Así como el mínimo entre el *complemento* de A a C y el *complemento* de B a C.

¿Qué tienes que entregar como resultado? Dos números, el cociente de la división de $A+B$ entre C y el mínimo de los *complementos*.

¿Dónde tienes que entregar el resultado? En las casillas $(1,1)$ y $(1,2)$ respectivamente.

Analiza el problema

¿Qué tienes que poder hacer para resolver el problema? 1. **Sumar:** Se requiere sumar valores para obtener el valor de $A+B$. 1. **Obtener el cociente de una división:** Se requiere para obtener $\frac{A+B}{C}$. 1. **Obtener el residuo de una división:** Se requiere para obtener el *complemento* de A a C y de B a C. 1. **Obtener el mínimo entre dos números:** Se requiere para quedarse con el menor de los *complementos*.

Este es un problema que tiene varios casos particulares que hay que cuidar para poder obtener todos los puntos. El problema consta de dos partes: obtener el cociente de la división y obtener el mínimo de los complementos.

Cuando te enfrentes a un problema dividido en más de una sección, resuelve una por una.

Cociente de la división.
Para obtener el cociente de la división $\frac{A+B}{C}$ se requiere hacer primero la suma y luego la división.

Ya que sumar es equivalente a *unir* o *agrupar*, la suma puede realizarse contando la cantidad de zumbadores en el primer montón y dejando una cantidad igual de zumbadores sobre el segundo montón.

▼ Haz clic para ver el código

```
define-nueva-instrucción suma como inicio
  si junto-a-zumbador entonces inicio
      coge-zumbador;
      suma;
      deja-zumbador;
  fin
  si-no avanza;
fin;
```

Para dividir un montón de zumbadores entre un número, digamos X, se puede intentar quitar montones de X zumbadores. Por cada montón que se pueda quitar hay que agregar 1 al cociente. Si al final, no se puede quitar un montón completo de X, no se debe agregar al cociente.

La dificultad de la idea anterior radica en determinar si pudiste quitar un montón de X zumbadores completo o no. Cuando se desea devolver un resultado del tipo "*Verdadero*" o "*Falso*" una técnica que suele ser útil es utilizar la orientación a forma de resultado. El siguiente código muestra un ejemplo de cómo hacerlo.

▼ Haz clic para ver el código

```
define-nueva-instrucción divide(x) como inicio
  repetir x veces inicio
      si junto-a-zumbador entonces coge-zumbador
      si-no sur;
  fin;
  si no orientado-al-sur entonces inicio
      divide(x);
      deja-zumbador;
  fin;
fin;
```

Con las dos `instrucciones` anteriores se puede resolver la primer parte de este problema.

***Complementos* de A y B a C**

Definamos como el *complemento* de un número A a un número C la cantidad que le falta a A para llegar a un múltiplo de C. Siendo más específicos *complemento* de A a C es igual a $kC - A$ donde k es el menor entero tal que $kC \geq A$.

Aunque no es necesario conocer esta definición para resolver el problema, tenerla bien definida hace que la implementación sea más sencilla.

¿Por qué se requiere el *complemento*? El problema especifíca que alguno de los hermanos le va a prestar al otro. Como se desea prestar lo mínimo posible, cuando un hermano le presta a otro le tiene que prestar una cantidad suficiente para llegar a un múltiplo de C (para poder comprar otro chocolate). Pero justo esa cantidad es lo que definimos como *complemento* (la cantidad necesaria para llegar al siguiente múltiplo de C). De modo que si podemos obtener el complemento de un número a otro, el resultado de la segunda parte del problema será el mínimo de ambos.

Para obtener el *complemento* podemos usar una variación de la instrucción `divide(x)` del código que aparece arriba. Esta vez lo que nos importa no es el cociente sino saber cuántos números nos faltó quitar del último grupo. El siguiente código muestra una forma de hacerlo.

▼ Haz clic para ver el código

```
define-nueva-instrucción complemento(x) como inicio
  mientras junto-a-zumbador y no orientado-al-sur hacer inicio
      repetir x veces inicio
          si junto-a-zumbador y no orientado-al-sur entonces coge-zumbador
          si-no inicio
              sur;
              deja-zumbador;
          fin
      fin;
  fin;
fin;
```

Hay muchas formas de obtener el mínimo entre dos números. El siguiente código muestra una forma de hacerlo

▼ Haz clic para ver el código

```
define-nueva-instrucción deja-minimo(x) como inicio
  si junto-a-zumbador y no si-es-cero(x) entonces inicio
      coge-zumbador;
      deja-minimo(precede(x));
      deja-zumbador;
  fin
  si-no coge-todo;
fin;
```

Con las instrucciones anteriores y un par de instrucciones auxiliares más se puede resolver el problema completo. Un posible pseudocódigo de la solución es:

- Saca una copia de A.
- Suma A y B en la posición $(1, 1)$.
- Aprende C y haz lo siguiente:
  - Divide $A + B$ entre C dejando el resultado en la posición $(1, 1)$.
  - Obtén el complemento de A a C.
  - Obtén el complemento de B a C.
  - Deja el mínimo de los complementos en la posición $(1, 2)$.
- Si el mínimo de los complementos es mayor o igual que la mitad de C, no dejes nada.

Este es un problema que tiene varios casos especiales, te recomiendo analizar con cuidado el código de la solución oficial.

▼ Haz clic para ver el código

```
iniciar-programa
  define-nueva-instruccion norte como mientras no orientado-al-norte hacer gira-izquierda;
  define-nueva-instruccion sur como mientras no orientado-al-sur hacer gira-izquierda;
  define-nueva-instruccion este como mientras no orientado-al-este hacer gira-izquierda;
  define-nueva-instruccion oeste como mientras no orientado-al-oeste hacer gira-izquierda;
  define-nueva-instruccion gira(n) como repetir n veces gira-izquierda;
  define-nueva-instruccion deja(n) como repetir n veces deja-zumbador;
  define-nueva-instruccion coge-todo como mientras junto-a-zumbador hacer coge-zumbador;

  define-nueva-instruccion copia(x) como inicio
    si junto-a-zumbador entonces inicio
        coge-zumbador;
        copia(sucede(x));
    fin
    si-no inicio
        deja(x);
        avanza;
        deja(x);
    fin;
  fin;

  define-nueva-instruccion deja-minimo(x) como inicio
    si junto-a-zumbador y no si-es-cero(x) entonces inicio
        coge-zumbador;
        deja-minimo(precede(x));
        deja-zumbador;
    fin
    sino coge-todo;
  fin;

  define-nueva-instruccion aprende-y-deja-minimo(x) como inicio
    si junto-a-zumbador entonces inicio
        coge-zumbador;
        aprende-y-deja-minimo(sucede(x));
    fin
    si-no inicio
        avanza;
        deja-minimo(x);
    fin;
  fin;

  define-nueva-instrucción divide(x) como inicio
    repetir x veces inicio
        si junto-a-zumbador entonces coge-zumbador
        si-no sur;
    fin;
    si no orientado-al-sur entonces inicio
        divide(x);
        deja-zumbador;
    fin;
  fin;

  define-nueva-instrucción complemento(x) como inicio
    mientras junto-a-zumbador y no orientado-al-sur hacer inicio
        repetir x veces inicio
            si junto-a-zumbador y no orientado-al-sur entonces coge-zumbador
            si-no inicio
                sur;
                deja-zumbador;
            fin
        fin;
    fin;
  fin;

  define-nueva-instruccion quita-si-mayor-que-mitad-de(x) como inicio
    si junto-a-zumbador y (no si-es-cero(x) y no si-es-cero(precede(x))) entonces inicio
        coge-zumbador;
        quita-si-mayor-que-mitad-de(precede(precede(x)));
        si no orientado-al-norte entonces deja-zumbador;
    fin
    si-no inicio
        si junto-a-zumbador entonces inicio
            coge-todo;
            norte;
        fin;
    fin;
  fin;

  define-nueva-instruccion procesaC(x) como inicio
    si junto-a-zumbador entonces inicio
        coge-zumbador;
        procesaC(sucede(x));
```

```
    fin
    si-no inicio

        avanza;
        avanza;
        divide(x);

        si junto-a-zumbador entonces inicio
            este;
            avanza;
            complemento(x);

            norte;
            avanza;
            complemento(x);
            sur;

            aprende-y-deja-minimo(0);
            quita-si-mayor-que-mitad-de(x);
        fin
        si-no inicio
            este;
            avanza;
            coge-todo;
            apagate;
        fin;
    fin;
  fin;

  inicia-ejecución
    copia(0);
    este;
    copia(0);

    sur;
    avanza;
    oeste;
    copia(0);

    este;
    avanza;
    avanza;
    oeste;

    procesaC(0);
    apágate;
  termina-ejecución
finalizar-programa
```