ACM-UCLA Programación Creativa 2013

Maratón Novel

23 de Febrero de 2012

Universidad Centroccidental Lisandro Alvarado

Este problemario contiene 7 problemas; las páginas están enumeradas de 1 a 7.

Problema A

Acuario

Nombre de archivo: `acuario.c`, `acuario.cpp` *o* `acuario.java`
Debe leer desde entrada estándar e imprimir a salida estándar.

Es bien conocido que en un acuario algunos peces se pueden comer a otros. Usted tiene un acuario que contiene una cantidad de peces del cual conoce el tamaño. Usted sabe que un pez se puede comer a otro, solo cuando está en el rango de entre el doble de tamaño o 10 veces más grande. Se quiere agregar un pez a la pecera, pero queremos determinar el tamaño para no causar conflictos de comerse con otros peces. Considerando esto usted debe escoger un pez que este entre los siguientes tamaños

- No hay riesgo de ser comido por otro pez si su tamaño no está entre 1/10 y 1/2 inclusive, del tamaño de otro pez.
- No tiene tentación de comerse a otro pez si el tamaño de los otros peces no están entre 1/10 y 1/2 inclusive de su tamaño.

Por ejemplo, si los tamaños de los peces están entre 1 y 12, se encuentra un pez de tamaño 1, y queremos insertar un pez, ese puede tener tres posibles tamaños. Los posibles tamaños para el pez que están fuera del rango establecido son 1, 11, 12.

Entrada

La entrada consiste de varias líneas. La primera línea de un caso de prueba consiste en el tamaño más pequeño. La segunda línea consiste en el tamaño más grande que puede tener. La tercera línea tiene el número de peces en el acuario. La cuarta línea tiene los tamaños de los peces del acuario separado por un espacio. La entrada finaliza con EOF.

Salida

Escriba en una línea el número de tamaños que puede hallar y que no causen conflictos entre peces.

Ejemplo de entrada

```
1
12
1
1
2
999
6
941 797 120 45 7 120

```

Ejemplo de Salida

```
3
10

```

Problema B

Bits

Nombre de archivo: `bits.c`, `bits.cpp` *o* `bits.java`
Debe leer desde entrada estándar e imprimir a salida estándar.

Las computadoras operan en números binarios. Casi todos los cálculos se realizan manipulando 0s y 1s. Para que las computadoras puedan utilizar los números que le damos, hay que convertirlos de la base 10 que normalmente tenemos, a la base binaria (2). En muchas ocasiones es útil determinar cuántos bits se requieren para representar un número, con la finalidad de ahorrar espacio. Por ejemplo cualquier número menor a 256 se puede representar con 8 bits.

Para hallar el equivalente decimal de un número binario procedemos como sigue: Para cada numero 1 sumamos las potencias 2^i donde i el numero de dígitos a la derecha del uno. Por ejemplo el equivalente decimal del número binario 10100 se halla como sigue: a la derecha del primer 1 hay 4 dígitos dando $2^4 = 16$, a la derecha del segundo 1 hay dos dígitos que representa $2^2 = 4$. Sumando ambos tenemos su equivalente decimal que es 20.

Entrada

La entrada contiene el numero que queremos representar en binario, la entrada termina cuando el numero es igual a 0.

Salida

Escriba en una línea el número mínimo de bits que se requiere para representar este número.

Ejemplo de entrada

```
32
12
1
0

```

Ejemplo de Salida

```
6
4
1

```

Problema C

Cadena

Nombre de archivo: `cadena.c`, `cadena.cpp` *o* `cadena.java`
Debe leer desde entrada estándar e imprimir a salida estándar.

Se te dará una lista de cadenas. Cada carácter de la entrada será calculado como sigue:

Clave = (Posición en el alfabeto) + (numero de elemento) + (posición en el elemento)

Todas las posiciones comienzan con 0, por ejemplo la letra A tiene la posición 0, la B la 1, así sucesivamente. La respuesta esperada es la suma de las claves de caracteres en la entrada. Por ejemplo si tenemos "`CBA DDD`", cada elemento seria calculado como sigue:

2 = 2 + 0 + 0: C en el elemento 0 posición 0
2 = 1 + 0 + 1: B en el elemento 0 posición 1
2 = 0 + 0 + 2: A en el elemento 0 posición 2
4 = 3 + 1 + 0: D en el elemento 1 posición 0
5 = 3 + 1 + 1: D en el elemento 1 posición 1
6 = 3 + 1 + 2: D en el elemento 1 posición 2

El resultado final será 2 + 2 + 2 + 4 + 5 + 6 = 21.

Entrada

La entrada consiste de varios casos de prueba. Cada caso de prueba viene en una línea que contiene todos los elementos separados por un espacio. La entrada termina cuando no hay más datos.

Salida

Por cada caso de prueba escriba en una línea la clave encontrada con el procedimiento anterior.

Ejemplo de entrada

```
1  CBA DDD
2  Z
3  A B C D E F
4  ABCDEFGHIJKLMNOPQRSTUVWXYZ
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
5  ZZZZZZZZZZ
6
```

Ejemplo de Salida

```
1  21
2  25
3  30
4  1326
5  295
6
```

Problema D

Rectángulo más Grande

Nombre de archivo: `rectangulo.c`, `rectangulo.cpp` *o* `rectangulo.java`
Debe leer desde entrada estándar e imprimir a salida estándar.

El pequeño José ha encontrado muchos palitos que son de 1 centímetro de largo. El quiere hacer un rectángulo con la mayor área posible, utilizando los palitos como perímetro.

Se le está permitido juntar dos palitos, pero no se le permite romper los palitos en unos más pequeños. Por ejemplo, si José tiene 11 palitos él puede construir un rectángulo de 2 x 3 utilizando 10 palitos. Este rectángulo tiene una superficie de 6 centímetros cuadrados, que es la mayor área que puede formar.

Dado el número de palitos, halle el área del rectángulo más grande que puede formar.

Entrada

La entrada viene en una línea que contiene el número de palitos que José encontró, se termina la entrada cuando el número de palitos es igual a 0.

Salida

Por cada caso de prueba escriba en una línea el área del rectángulo más grande que puede formar.

Ejemplo de entrada

```
11
5
64
753
7254
0

```

Ejemplo de Salida

```
6
1
256
35344
3288782

```

Problema E

Sumar Dígitos

Nombre de archivo: `cuti.c`, `cuti.cpp` *o* `cuti.java`
Debe leer desde entrada estándar e imprimir a salida estándar.

Comenzando con un entero entre 00 y 99 inclusive, escritos como dos dígitos (use un cero a la izquierda en caso de que el numero sea menor que 10). Realice lo siguiente:

1. Sume los dos dígitos.
2. Ahora concéntrese en el digito de la derecha, en el número original y en el de la suma.
3. Finalmente combine estos números.

Si repetimos este proceso varias veces obtenemos el número original.

Por ejemplo,

Inicio	Sume los dos dígitos	Combine los dos dígitos
26	2 + 6 = 08	'6' y '8' = 68
68	6 + 8 = 14	'8' y '4' = 84
84	8 + 4 = 12	'4' y '2' = 42
42	4 + 2 = 06	'2' y '6' = 26

En este caso tomo 4 pasos para obtener el número original. Se pide que devuelva el número de pasos requeridos para obtener el número original.

Entrada

Existe C casos de pruebas y luego, por cada caso de prueba, una línea que contiene un numero $0 \leq N \leq 99$.

Salida

Por cada caso de prueba escriba en una línea el número de pasos requeridos para obtener el número original.

Ejemplo de entrada

```
3
26
55
0

```

Ejemplo de Salida

```
4
3
1

```

Problema F

Gusanillo

Nombre de archivo: `gusanillo.c`, `gusanillo.cpp` *o* `gusanillo.java`
Debe leer desde entrada estándar e imprimir a salida estándar.

El gusanillo es una criatura de hábitos regulares. Se estira hacia adelante alguna distancia a lo largo de una rama de un árbol y para descansar. Si se detiene en una hoja se la come. Luego se estira la misma distancia que se estiro antes, y repite esta rutina hasta que se pasa del final de la rama.

Considere un gusanillo que recorre la longitud de una rama cuyas hojas esta espaciadas a intervalos uniformes. Dependiendo de la distancia entre el gusanillo y sus puntos de descanso puede o no puede comerse todas las hojas. Siempre existe una hoja al principio de la rama donde descansa antes de iniciar su recorrido.

Dados tres valores que especifican la longitud de la rama en centímetros, la distancia que recorre entres descansos y la distancia entre un par consecutivo de hojas, calcule el numero de hojas que el gusanillo consumirá.

Entrada

En una línea, separados por un espacio, viene el tamaño de la rama, la distancia que recorre entre descansos y la distancia entre pares consecutivos de hojas. La entrada termina cuando no hay más datos.

Salida

Escriba en una línea el número de hojas que se comerá el gusanillo.

Ejemplo de entrada

```
11 2 4
12 6 4
20 3 7
21 7 3
15 16 5

```

Ejemplo de Salida

```
3
2
1
2
1

```

Problema G

El Curioso Bob

Nombre de archivo: `bob.c`, `bob.cpp` *o* `bob.java`
Debe leer desde entrada estándar e imprimir a salida estándar.

Bob es un niño muy curioso, y un día cuando jugaba en el Río Piedra se sentó en la orilla y empezó a armar triángulos con las piedras que encontraba. Los triángulos tenían la misma cantidad de piedras por cada lado (era equiláteros, pero Bob aún no conocía esa palabra), de la siguiente forma:

Bob vio que si el triangulo tenía 2 piedras por lado, en total el triángulo llevaba 3 piedras. Luego contó las del segundo triángulo y vio que eran 6. Bob quiso saber cuántas piedras le llevaría armar un triángulo con cuatro o cinco piedras por lado, pero no encontró suficientes piedras (el Río Piedra en realidad es predominantemente arenoso, el que lo nombró era un bromista, al parecer).

Construye un programa que, dada la cantidad de piedras por lado, le diga a Bob cuántas piedras lleva en total el triángulo.

Entrada

La entrada consiste de varios casos de prueba. Cada caso de prueba comprende una línea con un entero L ($1 \leq L \leq 10^6$), la cantidad de piedras de cada lado del triángulo. La entrada finaliza con un caso de prueba donde $L = 0$. Este último caso no debe ser procesado.

Salida

Por cada caso de prueba, imprima una línea con un entero que indique la cantidad de piedras que lleva el triángulo en total.

Ejemplo de entrada

```
2
3
6
0

```

Ejemplo de Salida

```
3
6
21

```