Problema - Buscando el Diez

A usted le dan todas las notas que ha obtenido en una clase. Cada nota está entre 0 y 10 inclusive.

Asumiendo que en todas sus próximas tareas obtendrá un 10, determine el número de tareas necesarias para obtener un 10. Usted recibirá un 10 si su promedio es 9,5 o superior.

Por ejemplo si sus notas son $8, 9$ entonces requerirá 4 tareas adicionales en las que tendrá que sacar 10. Con cada tarea su promedio se aumentaría a $9, 9{,}25, 9{,}4$ y 9,5.

Entrada

La entrada esta en una línea que contiene todas las notas que obtuvo separadas por una coma.

Salida

Escriba en una línea el número de tareas que requiere para obtener un 10.

Ejemplos de entrada

```
9, 10, 10, 9
8, 9
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
10, 10, 10, 10
```

Ejemplos de salida

```
0
4
247
0
```

Problema

Para el dato de entrada siguiente escriba un programa que halle la respuesta.

```
7, 7, 10, 10, 4, 6, 4, 6, 0, 6, 7, 4, 6, 6, 9
```

La respuesta que debes entregar es:

```
101
```

Análisis y Solución

Para resolver este problema solo debe hallar cuantos 10 requiere para obtener el promedio deseado. El detalle esta en el código

Programa que resuelve el problema

```
1  #include <iostream>
2  #include <string>
3  #include <cstdlib>
4
5  using namespace std;
6
7  int main() {
8      string line;
9      getline( cin, line );
10     double promedio = 0.0, total = 0.0;
11     int contador = 0;
12     line = line + ", ";
13     int mat;
14     for( mat=0; line.length()>0; mat++ ){
15       total += atof( line.substr(0,line.find(',')).c_str() );
16       line = line.substr( line.find(' ')+1 );
17     }
18     promedio = total/(double)mat;
19     if( promedio >= 9.5 ) {
20       cout << 0 << endl;
21     } else {
22       while( promedio <9.5) {
23         total +=10.0;
24         mat++;
25         contador ++;
26         promedio = total/(double)mat;
27       }
28       cout<< contador << endl;
29     }
30 }
```

Problema - El Resto

El pequeño Arturo adora los números y el 9 es su número favorito. Cuando encuentra un número siempre calcula el residuo después de dividir por 9. Este es su número de la suerte.

Esta vez le dieron un número X con una longitud N que no contiene ceros. Se le pide hallar el residuo de la división por 9 de $superSuma(X)$.

Se define $super(X)$ como sigue: Dado un conjunto de S con posiciones de dígitos en X borramos dígitos en estas posiciones para obtener un $sub - numero$. La $superSuma(X)$ es la suma de todos los $sub - numeros$.

Por ejemplo si $X = 123$ entonces $superSuma(X) = 123 + 12 + 13 + 23 + 1 + 2 + 3 = 177$.

Calcular mentalmente el residuo de dividir por 9 de la $superSuma(X)$ no le resulta fácil a Arturo por lo que tu tarea es ayudarle a resolver este problema.

Entrada

Cada línea contiene una cadena con los dígitos de X.

Salida

Escriba el resto de dividir la $superSuma(X)$ por 9.

Ejemplos de entrada	**Ejemplos de salida**
123 8 11235813213455	6 8 7

Problema

Para el dato de entrada siguiente escriba un programa que halle la respuesta.

```
24816
```

La respuesta que debes entregar es:

```
3
```

Análisis y Solución

Veamos el ejemplo:

$$Super(123) = 123 + 12 + 13 + 23 + 1 + 2 + 3 = 177$$

Lo que deseamos hallar es:

$$Super(123) = 123 + 12 + 13 + 23 + 1 + 2 + 3 = 177\,\%9$$

Por las leyes de la aritmética

$$= (123\,\%9) + (12\,\%9) + (13\,\%9) + (23\,\%9) + (1\,\%9) + (2\,\%9) + (3\,\%9)$$

Como se utilizan los mismos dígitos en cada operando podemos separar los mismos para ver que:

```
 123 = 1 * 100 + 2 * 10 + 3 * 1
 12 =           1 * 10 + 2 * 1
 13 =           1 * 10 + 3 * 1
 23 =           2 * 10 + 3 * 1
  1 =                    1 * 1
  2 =                    2 * 1
  3 =                    3 * 1
```

Veamos la siguiente propiedad del

```
   1 % 9 =   1
  10 % 9 =   1 * (10 % 9) = (1 * 1) % 9 = 1
 100 % 9 = (10 % 9) * (10 % 9) = (1 * 1) % 9 = 1
1000 % 9 = (100 % 9) * (10 % 9) = (1 * 1) % 9 = 1
```

De aquí deducimos que:

```
Super(123) = ((1 + 2 + 3) + (1 + 2) + (1 + 3) + (2 + 3) + (1) + (2) + (3)) % 9
```

agrupando los digitos tenemos

```
Super(123) % 9 = ( 1 * 4 + 2 * 4 + 3 * 4 ) % 9
                 = ( 24 ) % 9
                 = 6 % 9
                 = 6
```

Programa que resuelve el problema

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main() {
7     string X;
8     cin >> X;
```

```
9       int n = X.length();
10      long long s = 0L;
11      for( int i=0; i<n; i++ ) {
12        s += (long long)(X.at(i) -'0');
13      }
14      s = s <<(long long)(n-1);
15      cout << (s%9) << endl;
16  }
```

Problema - Múltiplos

Le dan tres números enteros $min, max, factor$. Se le pide hallar cuantos números que están en el rango desde min hasta max son divisibles por $factor$.

Por ejemplo si $min = 0, max = 14, factor = 5$ tenemos los números $0, 5, 10$ que son divisibles por 5 así que el resultado esperado es 3.

Entrada

En cada línea vienen los valores $-1000000 \leq min, max \leq 1000000, 1 \leq factor \leq 1000$ separados por un espacio.

Salida

Escriba una linea con la cantidad de números divisibles por $factor$ que se encuentran en el rango.

Ejemplos de entrada	**Ejemplos de salida**

`0 14 5`
`7 24 3`
`-123456 654321 997`
`-75312 407891 14`
	`3`
	`6`
	`780`
	`34515`

Problema

Para el dato de entrada siguiente escriba un programa que halle la respuesta.

```
65456 456789 13
```

La respuesta que debes entregar es:

```
30102
```

Análisis y Solución

Solo nos piden recorrer un rango de números y contar cuantos son divisibles por otro.

Programa que resuelve el problema

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5   int min, max, factor;
6   int res, i;
```

```
 7      cin >> min >> max >> factor;
 8      res = 0;
 9      for( i = min; i <= max; i++ ) {
10        if( i%factor == 0 ) {
11          res++;
12        }
13      }
14      cout << res << endl;
15  }
```

Problema - Rectángulo mas grande

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 puede crear el 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.

Salida

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

Ejemplos de entrada	**Ejemplos de salida**
11	6
5	1
64	256
753	35344
7254	3288782

Problema

Para el dato de entrada siguiente escriba un programa que halle la respuesta.

```
621
```

La respuesta que debes entregar es:

```
24025
```

Análisis y Solución

Un rectángulo tiene 4 lados, si dividimos la cantidad de palitos a la mitad, con los que quedan podemos hacer dos lados del rectángulo. Si es una cantidad si nos queda una cantidad par ya tenemos los dos lados dividiendo por 2. Si es impar basta con hacer que un lado tenga un palo mas que el otro.

Programa que resuelve el problema

```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <math.h>
5
6
7  using namespace std;
8
9  int main(){
10   int N=0;
11         cin>>N)
12     N=N/2;
13       if(N%2==0)
14         cout << ((N/2)*(N/2));
15       else
16         cout << (((N+1)/2)*((N-1)/2));
17
18 }
```

Problema - Carga Cajas

Su empresa ha comprado un nuevo robot para cargar cajas. Su tarea es programar el robot para empacar los items en cajas para el embarque. El robot tienen muy poca memoria así que esta restringido a colocar los items en las cajas todos en la misma orientación.

Cada item es rectangular y solido con dimensiones $itemX * itemY * itemZ$. La caja también es rectangular con dimensiones $cajaX * cajaY * cajaZ$.

Los items se pueden acomodar en la caja en cualquier posición ortogonal. Esto significa que los lados de los items deben ser paralelos a los lados de las cajas. Solo se pueden colocar items completos en una caja. Su tarea es determinar el máximo número de cajas de items que puede colocar en una caja. Todos los items con la misma dirección. Por ejemplo si la caja es de $100x98x81$ y los items de $3x5x7$ entonces orientando los items para que sean $5x7x3$ permite acomodar 7560 items.

Entrada

Cada línea contiene 6 números que representan $cajaX, cajaY, cajaZ, itemX, itemY, itemZ$.

Salida

Escriba el máximo numero de cajas de items que puede colocar en una caja.

Ejemplos de entrada	**Ejemplos de salida**
`100 98 81 3 5 7`	`7560`
`10 10 10 9 9 11`	`0`
`201 101 301 100 30 20`	`100`
`913 687 783 109 93 53`	`833`
`6 5 4 3 2 1`	`20`

Problema

Para el dato de entrada siguiente escriba un programa que halle la respuesta.

```
115 113 114 3 5 7
```

La respuesta que debes entregar es:

```
13984
```

Análisis y Solución

Se tienen 6 formas de acomodar las cajas. Solo tenemos que calcular cuantas cajas podemos colocar en cada forma. Luego hallamos el máximo.

Programa que resuelve el problema

```
#include <iostream>
#include <vector>
#include <string>
#include <cmath>


using namespace std;
int calc(int a,int b,int c,int d,int e,int f) {

  return (a/d)*(b/e)*(c/f);
}

int main(){
  int boxX=0;
  int boxY;
  int boxZ;
  int itemX;
  int itemY;
  int itemZ;

    cin >> boxY;
    cin >> boxZ;
    cin >> itemX;
    cin >> itemY;
    cin >> itemZ;

    int maximo=0;
    maximo=max(maximo,calc(boxX,boxY,boxZ,itemX,itemY,itemZ));
    maximo=max(maximo,calc(boxX,boxY,boxZ,itemX,itemZ,itemY));
    maximo=max(maximo,calc(boxX,boxY,boxZ,itemY,itemX,itemZ));
    maximo=max(maximo,calc(boxX,boxY,boxZ,itemY,itemZ,itemX));
    maximo=max(maximo,calc(boxX,boxY,boxZ,itemZ,itemX,itemY));
    maximo=max(maximo,calc(boxX,boxY,boxZ,itemZ,itemY,itemX));
    cout <<maximo;
}
```