

Problema - Acuario

Es bien conocido que en un acuario algunos peces se pueden comer a otros. Usted tiene un acuario que contiene un 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: 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 esté 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 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 separados por un espacio.

Salida

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

Ejemplos de entrada

Ejemplo 1

1

12

1

1

Ejemplo 2

2

999

6

941 797 120 45 7 120

Ejemplos de salida

Salida del Ejemplo 1

3

Salida del Ejemplo 2

10

Problema

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

```
3
997
16
10 11 12 13 14 16 82 83 84 85 720 730 740 750 760 770
```

La respuesta que debes entregar es:

147

Análisis y Solución

La solución del problema es bastante sencilla:

Se debe tomar todos los tamaños de los peces desde el más pequeño que en nuestro programa llamaremos *tamMin* hasta el más grande que denominaremos *tamMax*.

Para cada uno de los tamaños de peces se verifica si algún pez se lo puede comer. Si no se lo puede comer contamos este tamaño como una solución.

Al final imprimimos cuantas soluciones hemos encontrado.

Programa que resuelve el problema

```
1 // pseudo codigo
2 //definimos las variables que utilizaremos
3 int n=0, tamMin=0,tamMax=0;
4 mientras (cin>>tamMin){
5     leer(tamMax);
6     leer(n);
7     int pez[n];
8     for (int i = 0; i<n;i++)
9         leer(pez[i]);
10    int i, j, respuesta=0;
11    for i=[tamMin; tamMax]; {
12        for j=[0,n)
13            si (come(i, pez[j]) || come(pez[j], i))
14                break;
15            si (j==n)
16                respuesta++;
17        }
18    imprimir(respuesta);
19 }
20 }
```

Problema - Hash

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

$$\text{clave} = (\text{Posicion en el alfabeto}) + (\text{n'umero de elemento}) + (\text{posicion en el elemento})$$

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 los caracteres en la entrada. Por ejemplo si tenemos *CBA, DDD*, cada elemento sería 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 sería $2 + 2 + 2 + 4 + 5 + 6 = 21$.

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.

Ejemplos de entrada

```
CBA DDD
Z
A B C D E F
ABCDEFGHIJKLMNOPQRSTUVWXYZ ABCDEFGHIJKLMNOPQRSTUVWXYZ
ZZZZZZZZZZ
```

Ejemplos de salida

```
21
25
30
1326
295
```

Problema

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

ES WU DD WC PP OM OU VW DT JL YN VM WI YI OA YE ZV HA RG BO WA YM NJ GJ WA RU

La respuesta que debes entregar es:

1364

Análisis y Solución

El problema consiste en recorrer una secuencia de palabras que vienen en una fila. Por ejemplo si las palabras son **CBA** **DDD** entonces los elementos son las palabra **CBA** es cero y el elemento 1 es **DDD**. Las posiciones de las letras según el orden del alfabeto es como sigue: $A = 1, B = 2, C = 3, D = 4 \dots$ La posición en cada elemento comienza con 0, en el elemento 0, las posiciones son: $C = 0, B = 1, A = 2$.

Con estas consideraciones resolvemos el problema.

Programa que resuelve el problema

```
1 // pseudocodigo
2
3 leer una linea completa
4 iniciar el resultado en 0
5 for elemento = (desde el primero, al ultimo) {
6   Sacar un elemento de la fila
7   for pos =(0, hasta el tamano de elemento){
8     resultado = resultado + caracter en la posicion pos + elemento +
      posicion;
9   }
10 }
11 imprimir (resultado);
```

Problema - Monitoreo de Azimuth

Un robot se está moviendo en el plano siguiendo un conjunto de instrucciones.

Las instrucciones permitidas son:

| | |
|-------------------|--|
| <i>LEFT</i> | doblar 90 grados a la izquierda |
| <i>RIGHT</i> | doblar 90 grados a la derecha |
| <i>TURNAROUND</i> | doblar 180 grados |
| <i>LEFTX</i> | doblar X grados a la izquierda, donde X es un número entero. |
| <i>RIGHTX</i> | doblar X grados a la izquierda, donde X es un número entero. |
| <i>HALT</i> | parar, no se ejecutan más instrucciones. |

Se pide hallar los grados en sentido al recorrido del reloj, en los que estaría apuntando el robot.
Por ejemplo si las instrucciones son:

- RIGHT 59
- RIGHT
- RIGHT
- HALT
- LEFT
- LEFT
- LEFT

Para hallar la posición del robot procedemos como sigue a la derecha 59 grados, derecha otra vez dando 149 grados, derecha otra vez 239. Después detener, por lo que el resultado es 239.

Veamos otro ejemplo:

- LEFT

En este ejemplo recorreremos 90 grados a la izquierda por lo que tendremos $360 - 90 = 270$

Entrada

La entrada consiste de varias líneas. La primera línea contiene el número N de instrucciones. Las siguientes N líneas tienen las instrucciones del robot una instrucción en cada línea.

Salida

Escriba los grados en sentido al recorrido del reloj, en los que estaría apuntando el robot.

| Ejemplos de entrada | Ejemplos de salida |
|--|-----------------------------|
| Ejemplo de entrada 1 1 RIGHT | Salida del ejemplo 1 90 |
| Ejemplo de entrada 2 3 LEFT | Salida del ejemplo 2 0 |
| Ejemplo de entrada 3 8 LEFT 5 RIGHT 10 LEFT 15 RIGHT 20 LEFT 25 RIGHT 30 LEFT 35 RIGHT 40 | Salida del ejemplo 3 20 |
| Ejemplo de entrada 4 7 RIGHT 59 RIGHT RIGHT HALT LEFT LEFT LEFT | Salida del ejemplo 4 239 |

Problema

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

```
22
LEFT 43
LEFT 10
TURN AROUND
LEFT 155
LEFT 87
RIGHT 62
RIGHT 94
LEFT
RIGHT 13
RIGHT
LEFT 69
LEFT 90
LEFT 177
```

```
RIGHT 157
LEFT 74
LEFT 106
LEFT 133
LEFT 142
HALT
LEFT 156
RIGHT 62
RIGHT 84
```

La respuesta que debes entregar es:

140

Análisis y Solución

La solución se halla haciendo un seguimiento de las instrucciones del robot.

En cada iteración hay que separar la instrucción de los grados y luego hacer la suma.

Programa que resuelve el problema

```
1 //psudoo codigo que resuelve el problema.
2 Leer la cantidad de instrucciones y guardar en n.
3 Comenzamos con res=0
4 Leer una linea de texto y almacenar en s
5 Ahora procesar la instruccion
6     if( s=="RIGHT" )
7         res += 90;
8     else if( s=="TURN_AROUND" )
9         res +=180;
10    else if( s=="HALT" )
11        break;
12    else if( s== "LEFT" )
13        res -= 90;
14    else {
15        Separar en texto y numero.
16        if( inst == "RIGHT" )
17            res += numero;
18        else
19            res -= numero;
20    }
21 }
22 res +=3600000;
23 cout << (res \% 360) << endl;
24 }
```


Problema - Juego Balanceado

Se trata de un juego en el cual cada elemento representa un jugador en un juego de múltiples jugadores. El elemento i representa el jugador i y el carácter j de cada elemento i representa si el jugador i va a ganar W , empatar t , o perder L contra el jugador J .

Su tarea es asegurar que cada jugador gane al menos al $p\%$ con los otros jugadores, y pierda al menos un $q\%$ con los otros jugadores. Comenzando en índice cero que corresponde al primer jugador, debe imprimir el índice del primer jugador que no cumple esta condición. Si todos los jugadores cumplen este requerimiento debe imprimir -1 .

Vea que si hay N jugadores al menos, el número de caracteres $\text{techo}((N - 1) * p/100)$ del elemento i debe ser W . La fórmula para las pérdidas es análoga.

Entrada

La entrada consiste de varias líneas. La primera línea contiene el número N de elementos. Luego siguen N elementos seguidos por un espacio. Después vienen los valores P y Q .

Salida

Escriba en una línea el índice menor para el cual no se cumple el requerimiento solicitado o -1 si ningún elemento cumple el requerimiento.

Ejemplos de entrada

Ejemplo de entrada 1

```
4
TWWW LTWW LLTW LLLT
20
20
```

Ejemplo de entrada 2

```
4
TWWW LTWW LLTW LLLT
0
```

```
0
```

Ejemplo de entrada 3

```
3
TTT TTT TTT
1
1
```

Ejemplos de salida

Salida para el ejemplo 1

```
0
```

Salida para el ejemplo 2

```
-1
```

Salida para el ejemplo 3

```
0
```

Problema

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

18

TLLLLLTWWWWTLLWWWT WTTWTTLLWLLWWLTLWW


```

WTTWWT LWTLW LWWWL TW WLLTL TWWTW LLLWLT
WTLWTL WWWWL LLLWLTW WTTTW TWLLW TLLWWWLW
TWLLLL TLLW TWWWL LW LWWWL TWLLW LLLWT
LLTLL WWTWLTW TLWT LWL TLL LWT TLL LLLW TW
LWWLW TTWLT TLLWLL TLLW WLLW TTW TTLL
WLLLW LLLTW WTWLLW WLLW WLLWL WLLTTWLL
LTLW WLTWLTW TTTWT LWLLL W WLLTWL TTLLW
LLTW TLLLL TWLWLTW TLL TLL TLLW LWTLLT
18
6

```

La respuesta que debes entregar es:

17

Análisis y Solución

Para resolver este problema hay que leer todos las cadenas con los datos de que ganó y perdió. Luego leemos p, q y calculamos los porcentajes

Calcular $\text{gana} = (\text{int}) \text{ceil}((N-1) * p / 100.0)$

Calcular $\text{pierde} = (\text{int}) \text{ceil}((N-1) * q / 100.0);$

Luego hallamos el numero de veces que ganó o perdió cada jugador. Esto se hace recorriendo la cadena carácter por carácter.

Finalmente si algún carácter esta entre gana y pierde, imprimimos el índice que representa el numero de jugador imprimimos el resultado y finalizamos.

Programa que resuelve el problema

```

1 //Psudo codigo que resuelve el programa
2
3 Leer $n$ el numero de jugadores.
4 Leer las $n$ cadenas i y guardar en un vector denominado conflictos.
5 Leer p y q
6 for i=(0,n) {
7   Calcular gana = (int) ceil( (N-1) * p / 100.0)
8   Calcular pierde = (int) ceil( (N-1) * q / 100.0);
9   numGana = 0, numPierde = 0;
10  // contar el numero de W y L.
11    for j=(0,j < conflictos[i].size())
12      char c = conflictos[i].at(j);
13    if (c == 'W') numGana++;
14    else if (c == 'L') numPierde++;
15    if (numGana >= gana && numPierde >= pierde) continue;
16    else {
17      result = i; break;
18    }

```

```
19     }  
20     imprimir result;  
21 }  
22 }
```

