ACM-UCLA Programación Creativa 2012

Práctica Uno

14 de Abril de 2012

Universidad Centroccidental Lisandro Alvarado

Este problemario contiene 8 problemas; las páginas están enumeradas de 1 a 10.

Problema A

Multitap

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

Un teclado telefónico está compuesto por doce teclas: una para cada número del 0 al 9, una para el símbolo asterisco (*) y otra para el símbolo numeral (#). Para escribir mensajes de texto usando estos teclados uno de los métodos más empleados es el método multi-tap (o multi-press), en el cual se le asignan letras del alfabeto inglés a cada número del 2 al 9 de la siguiente forma:

2 = ABC
3 = DEF
4 = GHI
5 = JKL
6 = MNO
7 = PQRS
8 = TUV
9 = WXYZ

Para escribir un mensaje usando este método, se siguen las siguientes reglas:

- Cada pulsación de la misma tecla recorre el arreglo de letras asignado a dicha tecla. Por ejemplo, para obtener la letra 'E' pulsamos dos veces seguidas la tecla 3; para obtener la letra 'Z' pulsamos cuatro veces seguidas la tecla 9; y así sucesivamente.
- El arreglo de letras asignado a cada tecla se recorre cíclicamente, por lo tanto si pulsamos la tecla 8 cuatro veces seguidas obtendremos la letra 'T'; y si pulsamos la tecla 5 seis veces seguidas obtendremos la letra 'L'.
- Cuando el usuario haga una pausa en el tecleo -que representaremos con el caracter punto ('.')- o pulse una nueva tecla se imprimirá la letra en que haya quedado el ciclo de recorrido de las letras de la tecla anterior; la próxima vez que se pulse dicha tecla se iniciará el ciclo desde la primera posición del arreglo de letras asignado a dicha tecla.

Además, consideraremos que la tecla "0" (número cero) se usa para generar un espacio en blanco. Cada vez que se presione esta tecla se generará un espacio en blanco, independientemente de si se presiona la tecla más de una vez consecutivamente.

Las teclas 1, asterisco (*) y numeral (#), no tendrán ninguna función en nuestro modelo y no serán usadas.

Entrada

La entrada está compuesta por varios casos de prueba. Cada caso de prueba está compuesto por una cadena no vacía que representa una secuencia compuesta por combinaciones de los dígitos 0, 2, 3, 4, 5, 6, 7, 8, 9, que representan pulsaciones de las correspondientes teclas, y caracteres punto (.), que representan pausas en el tecleo. Al final de cada secuencia siempre se encuentra una pausa.

El fin de la entrada corresponde al fin del archivo de entrada (EOF).

Salida

Por cada caso de prueba imprima una línea que contenga el mensaje de texto resultante de usar la secuencia de pulsaciones y pausas en un teclado telefónico empleando el método multitap. Todas las letras deben estar en mayúscula.

Ejemplo de entrada

```
2.2226.
7282.
44444666.55520566633.
222.262.

```

Ejemplo de Salida

```
ACM
PATA
HOLA JOE
CAMA

```

Problema B

Pingüinos

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

El señor sin cuello, charlatán como pocos, se ha encontrado un difícil problema de algoritmia, o así dice él. Resulta que, encantado con pingüinos del todo el mundo, ha considerado a uno de ellos como el más importante, sin embargo no recuerda cual es.

Entonces ha decidido anotar la altura de cada pingüino, numerados de 1 a N, y va a considerar a aquel más alto como el nuevo más importante. Pero no sabe cómo resolver el problema y te ha pedido tu ayuda.

Entrada

Existe varios casos, cada uno comienza con un entero N ($1 \leq N \leq 100$) seguido por N enteros entre 1 y 30, donde el entero en la posición i indica la altura del pingüino i. La entrada finaliza con $N = 0$, este caso no debe ser procesado.

Salida

Por cada imprima una línea con el número de identificación del pingüino más alto. De haber varios con la mayor altura, imprimir el de menor número de identificación.

Ejemplo de entrada

```
3
1 4 2
0

```

Ejemplo de Salida

```
2

```

Problema C

Números Atractivos

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

Droopy es un joven que le gusta mucho la computación, en especial las competencias de programación. Un día estaba estudiando la representación binaria de los números y quedó sorprendido al ver que estos podían ser expresados utilizando únicamente ceros y unos. Luego de estudiar por un par de horas concluyó que los números que poseen más bits encendidos son más atractivos que los demás, por ejemplo, el número binario 1101 (3 bits encendidos) es más atractivo que 1001 (2 bits encendidos).

Ahora se le solicita que elabore un programa que dado dos números en su representación decimal dé como resultado el más atractivo de ellos.

Entrada

La entrada consiste de una serie de casos de prueba y esta será finalizada por EOF. Cada caso consiste de un par de números enteros a, b en su representación decimal ($0 \leq a, b \leq 10^9$)

Salida

Por cada caso de prueba debe imprimir el número más atractivo en su representación decimal considerando las reglas establecidas por Droopy. Si ambos números tienen la misma cantidad de bits encendidos en su representación binaria entonces muestre el mayor de ellos.

Ejemplo de entrada

```
15 8
3 4
5 7

```

Ejemplo de Salida

```
15
3
7

```

Problema D

Indio Manaure

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

Hace unos pocos meses emigró a Barquisimeto una tribu indígena conocida como "La tribu de Manaure", cuyo cacique es el gran "indio Manaure". Lamentablemente son bastante pobres, no tienen empleo y todavía no han conseguido un buen terreno para cultivar. Indio Manaure, quien es muy inteligente, obtuvo una gran idea para conseguir dinero y así acabar con el hambre que azota a su tribu, esta consiste en dirigirse a la avenida 20 y llevar consigo cierta cantidad de cajas con tickets premiados, el cobrará 1Bs a cada persona que desee tomar un ticket y de ser ganador le obsequiará algún objeto artesanal que el mismo ha construido.

Indio Manaure cuenta con N cajas, N tickets con la frase "Has ganado" y otros N con la frase "Sigue intentado", y sabe que debe repartir los tickets en las N cajas de forma tal que pueda obtener ganancias (No le conviene que las personas ganen). Además, no quiere dejar cajas vacías porque nadie las tomaría en cuenta. También deben saber que indio Manaure no cuenta con muchos objetos artesanales, así que en este momento no está interesado en saber cómo repartir los tickets en las cajas, sino en conocer cuál es la mayor probabilidad de NO tener que darle uno de estos objetos a alguien que seleccione un ticket.

Entrada

La primera línea contiene un entero $T < 50$ que indica el número de casos de prueba. Cada caso consiste de un entero N, $0 < N \leq 10^3$ el cual fue explicado anteriormente.

Salida

Por cada caso de prueba imprima la mayor probabilidad de que el jugador tome un ticket con la frase "Siga intentando". Imprimir 8 cifras decimales.

Ejemplo de entrada

```
1
1

```

Ejemplo de Salida

```
0.50000000

```

Explicación del caso de prueba: Tenemos 1 caja, un ticket que dice "Has ganado" y otro "Sigue intentado". Colocamos ambos ticket en la única caja que tenemos. El jugador sólo puede seleccionar una caja, y de esta 1 de los 2 tickets, así que la probabilidad de que el jugador tome el ticket "Sigue intentando" es ½ = 0.5

Problema E

Criptoanálisis

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

ACM (Asociación de Criptología Marciana) está estudiando como una antigua civilización del planeta Marte encriptaba sus mensajes para protegerlos contra sus enemigos, esto es muy complejo y difícil por tanto tu no tomaras parte de esta tarea, pero si necesitan de tus servicios para unas tareas menores para ayudar a su causa.

Tu tarea es tomar un texto e indicar cuáles y cuantos caracteres hay.

Entrada

La entrada empieza con un entero N, y seguidamente N líneas de texto (caracteres en mayúscula y minúscula, números y símbolos).

Salida

Deberás imprimir cada caracteres seguido de cuantas veces aparece en el texto, ordenado de menor a mayor dependiendo del valor ASCII de cada carácter. Tú debes ignorar los símbolos, solo concéntrate en mostrar los caracteres de la a-z, A-Z y 0-9.

Ejemplo de entrada

```
2
Hola, soy 1 texto de entrada 555!
Y yo tambien!

```

Ejemplo de Salida

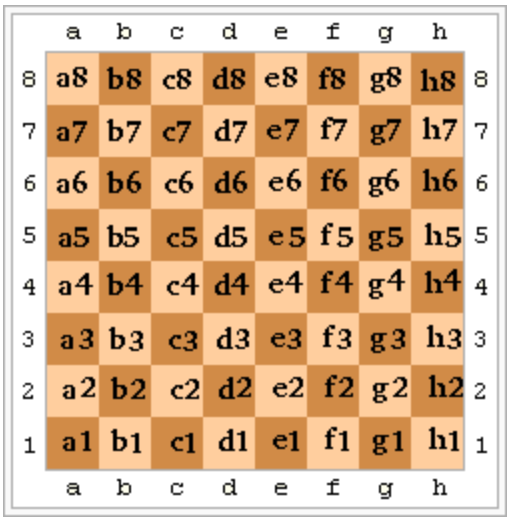
```
1 1
5 3
H 1
Y 1
a 4
b 1
d 2
e 4
i 1
l 1
m 1
n 2
o 4
r 1
s 1
t 4
x 1
y 2

```

Problema F

Ajedrez Jedi

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



Eres un maestro de Ajedrez, es tu pasatiempo favorito, pasas horas planeando estrategias y observando jugadas para mejorar cada día. En miras de tu entrenamiento para el Mundial Amateur de Ajedrez has decidido probar una nueva técnica basada en el movimiento de los caballos, pero esto se ha convertido en una ardua tarea y decidiste escribir un programa para ayudarte, pero lo perdiste así que debes escribirlo de nuevo.

Esta tarea es muy fácil porque ya la has hecho antes. El problema consiste en dado una posición *A* en un tablero de ajedrez, debes decir cuánto es el número mínimos de movimientos para llegar a una posición *B* solo moviéndote como un caballo, tu puedes estar seguro que siempre podrás llegar a la posición *B* aplicando estos movimientos.

Entrada

Un entero *N* indicando el número de casos, seguido de *N* líneas que indican cada caso de prueba. Cada línea contiene la posición *A* y la posición *B*.

Salida

Por cada caso imprima el número requerido en una línea.

Ejemplo de entrada

```
3
F6 F6
A1 H8
B2 E7

```

Ejemplo de Salida

```
0
6
4

```

Problema G

Problema de Bits I

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

Cada entero puede ser representado en un sistema binario, tenemos a 5 en base 10 corresponde a 101 en base 2. Y podemos definir como cantidad de bits activos al total de dígitos 1 que aparecen en la representación binaria de un número en particular, en este ejemplo 5 tiene dos bits activos.

El problema consiste en, dado dos enteros A y B, determinar el número X que se encuentre en el rango entre A y B, inclusive, con mayor cantidad de bits activos. De haber un empate entre dos o más números, buscar el menor de ellos.

Entrada

La entrada comienza con un entero T ($T \leq 100$) que indica el número de casos. Siguen T líneas, cada uno con un par de enteros A y B ($0 \leq A \leq B \leq 10^4$).

Salida

Por cada caso imprimir una línea con el valor de X.

Ejemplo de entrada

```
5
0 10
3 4
5 6
0 0
1 100

```

Ejemplo de Salida

```
7
3
5
0
63

```

Problema H

Repostaje

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

En un futuro lejano, años distantes de nuestra época, la tecnología ha avanzado a niveles inimaginables, y aunque los medios de teletransportación son la nueva boga, existentes aún están los antiguos medios de transporte, vehículos aéreos que trabajan mediante nuevas formas de energía.

Sin embargo, el repostaje de combustible aún sigue siendo un problema. La sobrepoblación y por ende, exceso de vehículos, han sobrepasado los límites de las estaciones. Un conjunto de científicos han vuelto del futuro solo para consultarte sobre el cómo solucionar el problema. Ellos quieren preparar una estación de abastecimiento donde se atenderá a cada vehículo en función del combustible restante, aquellos que posean menor carga tendrán mayor prioridad para ser atendidos, de haber un empate se atenderá a aquel que requiera menor tiempo de atención. La estación solo puede atender a un vehículo a la vez, y una vez que comienza con un vehículo no se detiene hasta atenderlo por completo. De no haber vehículos se queda en espera de la llegada de nuevos ingresos. Notar que mientras no sean atendidos tales vehículos, estos seguirán gastando combustible, a un ritmo de una unidad por minuto.

Tu trabajo consiste en indicar si es posible atender a todos los vehículos siguiendo el esquema propuesto sin que estos se queden sin combustible, dado situaciones a simular con una lista de ingreso con el momento de entrada, en minutos, la cantidad de combustible restante, y el tiempo que requieren de atención, también en minutos.

¿Puedes ayudar a resolver los problemas del futuro?

Entrada

La entrada contiene una serie de casos, en cada caso la primera línea contiene N ($1 \leq N \leq 10^5$), el número de vehículos a procesar, luego siguen N líneas, cada una con tres enteros T_i, C_i y R_i ($1 \leq T_i, C_i, R_i \leq 10^5$) indicando el ingreso de un vehículo i ($1 \leq i \leq N$) al minuto T_i y con combustible C_i, y requiere R_i minutos para ser atendido, los vehículos aparecen en orden no decreciente según su tiempo de ingreso. El último caso es seguido por una línea $N = 0$, este caso no se debe procesar.

Salida

Por cada caso, imprimir en una línea el mensaje "Y" si es posible atender sin que un vehículo posea el combustible vacío al momento de ser atendido, "N" en el caso contrario, sin comillas para ambos mensajes.

Ejemplo de entrada

```
3
1 5 5
4 10 3
6 4 4
2
1 5 10
2 9 3
4
1 3 5
6 4 5
6 4 3
30 3 5
2
1 8 5
2 3 4
2
500 8 10
505 3 8
0

```

Ejemplo de Salida

```
Y
N
Y
N
N

```