Universidad Andrés Bello
Facultad de Ingeniería
Ingeniería Civil Informática

Fundamentos de Programación
Solemne II

Nombre: _____ **Nota:** _____

Prof. Cátedra: Carlos Contreras Bolton **Fecha:** 23 de Octubre de 2014
Profs. Laboratorio: Felipe Reyes – Tamara Saéz – Nicolás Venegas – Omar Opazo

Instrucciones:

- Coloque su nombre a todas las hojas.
- Apague o silencie sus celulares. NO se podrá contestar llamadas ni visualizar el celular. Si realiza alguna de las acciones anteriores obtendrá nota mínima.
- Toda copia o intento de copia será calificada con nota mínima.
- Tiene 90 minutos para realizar la prueba.
- Tiene que realizar 3 de 4 preguntas.

1	
2	
3	
4	
Suma:	
	+1
Puntaje:	

Pregunta 1 (20 puntos)

Implementar la función `quitar_espacios(char cadenas[][100])` la cual recibe un arreglo de cadenas de caracteres. Debe imprimir todas las cadenas del arreglo en la que las secuencias de blancos de la cadena original, espacios en blanco (" ") y tabuladores ("\t") se sustituyan por un solo espacio en blanco.

Si, por ejemplo, en una posición i del arreglo tiene "una cadena con blancos ", mostrar por pantalla la cadena "una cadena con blancos ".

Pregunta 2 (20 puntos)

En un nuevo sistema de comunicación que ha ideado UNASoft todos los mensajes deben de ir encriptados. La idea base de esta nueva encriptación es colocar el mensaje adentro de otras palabras. Entonces, cada palabra que se quiere enviar se encuentra contenida en la palabra siguiente (puede ser como sub cadena o como las letras independientes en el orden de aparición).

Entrada

Cada caso de prueba contiene dos palabras, la primera corresponde a la del mensaje, mientras que la segunda a la encriptada. Se encuentran separadas por coma. Se sabe que la palabra tanto normal como encriptada tienen un máximo de 50 caracteres ASCII.

Salida

Cada caso de prueba debe indicar con un "`Si`" si la palabra se encuentra encriptada de esta forma y "`No`" si no está encriptada.

En la Tabla 1 se observa un ejemplo de entrada y salida. Debe realizar un programa en C.

Tabla 1: Ejemplo de entrada y salida.

Entrada	Salida

`sequence subsequence`
`person compression`
`VERDI vivaVittorioEmanueleReDiItalia`
`caseDoesMatter CaseDoesMatter`
	`Si`
	`No`
	`Si`
	`No`

Pregunta 3 (20 puntos)

En las carreras de autos, siempre hay un *pole position* junto a la línea de meta de la pista. Antes de que comience la carrera, el *pole* se utiliza para mostrar la grilla de salida. El número del primer auto en la grilla se muestra en la parte superior del *pole* y por debajo se muestra el número del auto en segundo lugar, y así sucesivamente.

Durante la carrera, el *pole* se utiliza para mostrar la posición actual de cada auto: el auto que está ganando la carrera tiene su número que aparece en la parte superior del *pole*, seguido por el auto que está en segundo lugar, y así sucesivamente.

Además de mostrar la posición actual de un auto, el *pole* también se utiliza para mostrar el número de posiciones de los autos que han ganado o perdido, en relación con la grilla de salida. Esto se hace al mostrar, de lado a lado con el número del auto, un número entero. Un valor v positivo al lado del número del auto en el *pole* significa que el auto ha ganado contra posiciones relativas a la grilla de salida. Un valor v negativo significa que auto ha perdido contra posiciones relativas a la grilla de salida. Un cero junto a un número de autos en el *pole* significa que el auto no ha ganado ni perdido ninguna posición con respecto a la grilla de salida (el auto está en la misma posición en la que comenzó).

Estamos en el centro del Gran Premio UNAB, la última carrera del Campeonato del universitario de autos. El director de la carrera, el Dr. Son Gokú, está preocupado: han habido algunas quejas de que el software que controla el sistema *pole position* es defectuoso, que muestra la información que no refleja el orden de carrera verdadera. Dr. Son Gokú ideó una manera de comprobar si el sistema de *pole* funciona correctamente. Dada la información que aparece en el *pole*, quiere reconstruir la grilla de salida de la carrera. Si es posible reconstruir una grilla de salida válida, que planea cotejarla con la parrilla de salida real. Por otro lado, si no es posible reconstruir una grilla de salida válida, el sistema de *pole* es de hecho defectuoso. ¿Puedes ayudar al Dr. Son Gokú?

Entrada

La entrada contiene varios casos de prueba. La primera línea de un caso de prueba contiene un entero N indicando el número de autos en la carrera ($2 \leq N \leq 10^3$). Cada uno de los siguientes N líneas contiene dos enteros C y P, separados por un espacio, que representan, respectivamente, el número del auto ($1 \leq C \leq 10^4$) y el número de posiciones que el auto ha ganado o perdido con respecto a la grilla de salida ($-10^6 \leq P \leq 10^6$), de acuerdo con el sistema de *pole*. Todos los autos en una carrera tienen diferentes números.

El final de la entrada se indica por una línea que contiene solamente un cero.

Salida

Para cada caso de prueba en la entrada, su programa debe imprimir una sola línea, que contiene la grilla de salida reconstruida, con el número de automóviles separados por espacios. Si no es posible reconstruir una parrilla de salida válida, la línea debe contener solamente el valor -1.

Deberá realizar su programa en C. La entrada es por teclado y la salida por pantalla. En la Tabla 2 muestra un ejemplo de entrada.

Tabla 2: Ejemplo de entrada y salida.

Entrada	Salida
4 1 0 3 1 2 -1 4 0 4 22 1 9 1 13 0 21 -2 3 19 1 9 -345 17 0 7 2 2 8 0 5 -2 7 1 1 1 9 1 3 -3 0	1 2 3 4 -1 -1 5 8 2 3 7 1 9

Pregunta 4 (20 puntos)

Para una matriz cuadrada de $n \times n$, realizar un programa sume los elementos punto a punto formando un reloj de arena, es decir, el elemento ubicado en $(0,0)$ se sumará con el $(n-1,0)$, el $(0,1)$ con el $(n-1,1)$ y así sucesivamente hasta completar la fila. Los elementos siguientes a sumar corresponderán a el $(1,1)$ con el $(n-2,1)$, hasta el $(1,n-1)$. Así sucesivamente hasta formar un triángulo en la parte superior y en la parte inferior de la matriz parecido a un reloj de arena. En otras palabras, como se ve en el siguiente ejemplo, se suman los elementos 1, los 2, los 3, etc.

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ * & 6 & 7 & 8 & * \\ * & * & 9 & * & * \\ * & 6 & 7 & 8 & * \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

Cada suma parcial deberá mostrarse por pantalla (cada suma en una línea).

La matriz es ingresada por teclado, en donde la primera línea contiene el tamaño de la matriz (n) y las siguientes n líneas restantes contiene la matriz de enteros. Cada elemento de la misma fila es separado por un espacio.

En la Tabla 3 se observa un Ejemplo de entrada y salida. Debe crear las siguientes funciones:

- Llenar una matriz.
- Realizar los cálculos.
- Imprimir por pantalla.

Tabla 3: Ejemplo de entrada y salida.

Entrada	Salida
5 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5	2 4 6 8 10 14 16 18 6