

Maratón de Programación - 2014 - Semifinal

CATEGORÍA Grado 10

30/10/2014

Politécnico Colombiano Jaime Isaza Cadavid



**Reglas de la competencia**

**Restricciones:**

1. Los equipos participantes no pueden utilizar ningún tipo de material digital para la realización de los ejercicios.
2. Ninguno de los miembros del equipo puede hablar con los competidores de los otros equipos participantes.
3. Los equipos no pueden utilizar a internet (Navegar) en ningún momento.
4. Cada uno de los integrantes del equipo debe apagar sus equipos móviles (Celulares, Palm’s, portátiles u otros similares).
5. Los miembros de equipo solo pueden usar el computador asignado un lápiz o lapicero y hojas en blanco.

**Método de trabajo:**

1. A cada equipo se le asignará de manera aleatoria uno de los computadores.
2. Los participantes recibirán instrucciones precisas de la forma de trabajo y las reglas del concurso.
3. Cada uno de los equipos tiene aproximadamente 4 horas para completar la prueba.
4. Los ejercicios pueden ser resueltos en diferente orden.
5. Cada ejercicio tiene igual puntuación.
6. Los equipos se clasificarán por tiempo, es decir se suman los tiempos desde el inicio de la prueba hasta la solución de cada ejercicio y se organizan los tiempos de menor a mayor.
7. Si alguno de los equipos termina correctamente (y en el menor tiempo) la solución de todos los ejercicios antes de las 4 horas planeadas se convierte inmediatamente en el ganador.

Los jurados y docentes tienen plena libertad de sancionar con la descalificación cualquier conducta irregular en el desarrollo de la prueba.



**ÉXITOS!!!**

**RETO 1**

**LA PALABRA CRUZADA**

**Nombre del archivo fuente:** cruzadas.c, cruzadas.cpp, o cruzadas.java

Desarrolle un programa que pida una palabra de tres (3) a 13 caracteres. Con esa palabra se va a formar una X en donde el carácter del medio se repite una sola vez. El programa debe validar que la cantidad de caracteres entrados sea impar y que no sea menor de 3 caracteres ni mayor de 13. De no cumplirse lo anterior el programa mostrará un mensaje según las indicaciones de los datos de salida.

**Datos de entrada** (entrada estándar):

Palabra impar de 3 a 13 caracteres en mayúsculas o minúsculas

**Datos de salida** (salida estándar):

* Si la palabra es de número impar de caracteres y tiene 3 o más caracteres y 13 o menos caracteres, hace una impresión en **mayúsculas** formando una x con la palabra, repitiendo sus letras menos la del medio y rellenando con guiones (-).
* Si la palabra es de número par de caracteres, hacer una impresión que diga:

**Palabra par. Intente de nuevo.**

* Si la palabra es menor de 3 caracteres, hacer una impresión que diga:

**Palabra menor de 3 caracteres. Intente de nuevo.**

* Si la palabra es mayor de 13 caracteres, hacer una impresión que diga:

**Palabra mayor de 13 caracteres. Intente de nuevo.**

**Ejemplo 1**

**amor**

Palabra par. Intente de nuevo.

**Ejemplo ejecución 2.**

**parangaricutirimicuaro**

Palabra mayor de 13 caracteres. Intente de nuevo.

**Ejemplo ejecución 3.**

**Linux**

L---L  
-I-I-  
--N--  
-U-U-  
X---X

**Ejemplo ejecución 4.**

**MicroSofT**

M-------M  
-I-----I-  
--C---C--  
---R-R---  
----O----  
---S-S---  
--O---O--  
-F-----F-  
T-------T

**RETO 2**

**DOBLES**

**Nombre del archivo fuente:** dobles.c, dobles.cpp, o dobles.java

Como parte de un programa de competencia aritmética, los estudiantes recibirán las listas generadas al azar de 2 a 15 números enteros positivos y únicos para determinar cuántos elementos de cada lista son el doble de otro elemento en la misma lista. Se necesita un programa que le ayude con la clasificación. Este programa debe ser capaz de analizar las listas y enviar la respuesta correcta para cada uno.

Por ejemplo, dada la lista 1 4 3 2 9 7 18 22 0

El programa debe responder 3,

El 2 es el doble de 1, 4 es el doble de 2, y 18 es el doble 9.

**Datos de entrada** (entrada estándar):

La entrada constará de una o más listas de números. Habrá una lista de números por línea. Cada lista contiene 2 a 15 números enteros positivos únicos. Ningún entero será mayor que 99. Cada línea se termina con el 0, que no se considera parte de la lista. Una línea con el número único -1 marcará el final de la entrada. El ejemplo de la siguiente entrada muestra tres listas separadas. Algunas listas pueden no contener dobles.

**Datos de entrada** (salida estándar):

La salida consistirá en una línea por cada lista de entrada, que contiene el número de elementos que son el doble de algún otro elemento.

| **Ejemplo Entrada** | **Ejemplo Salida** |
| --- | --- |
| 1 4 3 2 9 7 18 22 0 2 4 8 10 0  7 5 11 13 1 3 0  -1 | 3 2 0 |

**RETO 3**

**EPISODIO PERDIDO**

**Nombre del archivo fuente:** perdido.c, perdido.cpp, o perdido.java

Polycarpus adora las seriee de televisión. En este momento él está listo para terminar de ver una temporada de una popular comedia "Teoría de Grafos".  En total, la temporada tiene *n* episodios, numerados con números enteros de 1 a *n* .

Polycarpus tiene una lista de los episodios uno por uno, pero en un orden aleatorio.  Él ya ha visto todos los episodios a excepción de uno. ¿Qué episodio ha Polycaprus olvidado de ver?

**Datos de entrada** (entrada estándar):

La primera línea de la entrada contiene entero *n* (2 ≤  *n*  ≤ 100,000)  - el número de episodios en una temporada.  Suponga que los episodios están numeradas por números enteros de 1 a *n* .

La segunda línea contiene *n*  - 1 entero *un*1 ,  *un*2 , ...,  *un n* (1 ≤  *un i*  ≤  *n* )  - el número de episodios que Polycarpus ha visto. Todos los valores de *un i* son distintos.

**Datos de salida** (salida estándar):

Imprimir el número del episodio que Polycarpus no ha visto.

**Ejemplo**

**Entrada**

10   
3 8 10 1 7 9 6 5 2

**Salida**

4

**RETO 4**

**EL PEQUEÑO ELEFANTE**

**Nombre del archivo fuente:** elefante.c, elefante.cpp, o elefante.java

**EL PEQUEÑO ELEFANTE**

El pequeño elefante ama mucho su tierra. Su mayor amor está en el pueblo de “Elefantia”.

Sin embargo, Elefantia es muy pequeño para establecerse en el, por lo que el Pequeño Elefante quiere ir a alguna otra ciudad. El pequeño elefante no le gusta pasar mucho tiempo de viaje, por lo que para su viaje va a elegir una ciudad a la que necesite un tiempo mínimo para llegar. Si hay múltiples ciudades con la misma condición, el pequeño elefante no se va a ninguna parte.

Para cada pueblo, excepto Elefantia, se sabe el tiempo necesario para viajar a esta ciudad. Buscar la ciudad a la que el pequeño elefante irá a o imprimir "Me quedo en Elefantia", si se queda en Elefantia.

**Datos de entrada** (entrada estándar):

La primera línea contiene un entero **n** (1 ≤ **n** ≤ 105) - el número de ciudades. La siguiente línea contiene **n** enteros, separados por espacios: el i-ésimo entero representa el tiempo necesario para ir del pueblo Elefantia a la i-ésima ciudad. Los valores de tiempo son números enteros positivos, que no exceden de 109.

Se puede considerar a las ciudades numerados del 1 al n, ambos inclusive. Elefantia no se encuentra entre las ciudades numeradas.

**Datos de salida** (salida estándar):

Imprimir la respuesta en una sola línea - el número de la población del pequeño elefante irá. Si hay varias ciudades con mínimo tiempo de viaje, imprime "Me quedo en Elefantia " (sin las comillas).

**Ejemplos**

**Entrada**

2  
7 4

**Salida**

2

**Entrada**

7  
7 4 47 100 4 9 12

**Salida**

Me quedo en Elefantia

**Nota**

En el primer ejemplo sólo hay dos ciudades donde el pequeño elefante puede ir. El tiempo de viaje hasta la primera ciudad es igual a 7, a la segunda es 4. La ciudad más cercana a Elefantia es la segunda, por lo que la respuesta es 2.

En el segundo ejemplo las ciudades más cercanas son las ciudades de dos y cinco, el tiempo de viaje de las dos es igual a 4, por lo que la respuesta es "me quedo en Elefantia".