XI Concurso de Programación de la UAM
Primera fase: Examen calificatorio

División de Ciencias Básicas e Ingeniería
UAM Azcapotzalco

26 a 31 de mayo de 2014

¡Bienvenido!

El Concurso de Programación de la UAM tiene el propósito principal de fomentar el aprendizaje de la programación y los algoritmos entre los alumnos de las diversas licenciaturas de la UAM. Además, los alumnos participantes en el XI Concurso de Programación de la UAM tendrán la oportunidad de representar a nuestra institución en la etapa nacional del prestigioso evento ACM International Collegiate Programming Contest, a celebrarse en noviembre de 2014 en Querétaro. Los tres equipos ganadores de la etapa nacional representarán a México en la etapa mundial del mismo que se celebrará en Phuket, Tailandia en 2015.

El XI Concurso de Programación de la UAM se realizará en tres fases:

Primera fase Examen calificatorio (26 a 31 de mayo de 2014).

Segunda fase Examen eliminatorio (6 a 9 de junio de 2014).

Tercera fase Examen final (septiembre de 2014).

Gilgamesh

Todos los problemas del XI Concurso de Programación de la UAM estarán basados libremente en la antigua historia sumeria de Gilgamesh, el rey de Uruk. La primera parte de la historia trata sobre las aventuras de Gilgamesh y su amigo Enkidu, en las que derrotan a Humbaba y al toro celeste, pero que culminan con la muerte de Enkidu. En la segunda parte de la historia se relata el viaje de Gilgamesh en busca de la inmortalidad. Existen varias traducciones de esta historia al español, así que te invitamos a leerla.

Instrucciones

Aquí encontrarás los enunciados de tres problemas. Cada problema tiene un valor de 40 puntos, los cuales se obtendrán de las salidas que entregue tu programa para cada una de 40 entradas distintas. Todos los programas se evaluarán en Linux y se ejecutarán un máximo de 1 segundo. Que tu programa funcione con el ejemplo no quiere decir que funcionará siempre. Deberás enviar por correo electrónico el código fuente de los problemas que resuelvas a la dirección `uam14acm@gmail.com` indicando claramente los siguientes datos:

Datos personales Nombre completo, fecha de nacimiento y teléfono.

Datos académicos Número de matrícula y créditos aprobados.

Otros Licenciatura y Unidad (Azcapotzalco, Cuajimalpa o Iztapalapa).

De preferencia envía todos tus códigos en un solo correo. El nombre de los archivos que envíes debe ser de la forma `prg.zzz`, donde `prg` es el nombre del programa fuente y `zzz` es la extensión, según el lenguaje que hayas usado. Si así lo deseas, puedes resolver cada problema en un lenguaje distinto. Tu programa deberá leer e imprimir exactamente los datos que se indican (ni más ni menos) usando la entrada y la salida estándar. En particular, tu programa no deberá borrar la pantalla, escribir ningún tipo de letrero adicional, usar la biblioteca `conio`, hacer pausa, etc.

Quienes programen en C deberán usar `gcc`, quienes programen en C++ deberán usar `g++` y quienes programen en Java deberán usar `gcj` como sigue:

C `gcc prg.c -o prg -lm`

C++ `g++ prg.cpp -o prg -lm`

Java `gcj prg.java -o prg`

Deberás enviar tus códigos fuente y los datos solicitados a la dirección de correo `uam14acm@gmail.com` a más tardar el 31 de mayo de 2014 a las 22:00.

Entrada, salida y evaluación

La mayoría de los programas requieren leer datos proporcionados por el usuario y emiten una salida con el resultado de dicho programa. Existen muchas maneras en las que un programa puede leer y emitir datos; la manera por defecto es mediante la *entrada* y la *salida estándar*.

La entrada y salida estándar son flujos de datos que están disponibles automáticamente. Dichos flujos están usualmente vinculados a la consola. Los programas en C pueden usar la entrada y la salida estándar con las funciones de biblioteca `scanf` y `printf` respectivamente (además de algunas adicionales, como `getchar` y `putchar`). Los programas en C++ pueden usar las mismas utilidades que C y algunas adicionales: el objeto `cin` está vinculado por defecto a la entrada estándar y el objeto `cout` a la salida estándar. De manera similar, los programas en Java pueden usar los objetos `System.in` y `System.out` para la entrada y salida de datos.

Para calificar los programas que envíes al concurso, se comparará lo que tu programa haya emitido en la salida estándar con lo que emita un programa correcto usando los mismos datos de entrada. La comparación será literal, es decir, la salida de tu programa y la salida del programa correcto deben ser *idénticas*. Es por esto que debes evitar imprimir cosas adicionales a lo especificado en cada problema: ya que nuestro programa no imprimirá cosas adicionales, tu salida será diferente si tu programa sí lo hace.

La mayoría de los sistemas operativos permiten vincular la entrada y salida estándar de un programa a algo que no sea la consola de comandos. Es posible, por ejemplo, vincular dichos flujos de datos a archivos. Esto es muy útil cuando tu programa usa `scanf` y `printf` pero quieres guardar la salida de tu programa en un archivo o cuando quieres evitar tener que escribir la entrada manualmente cada vez que ejecutes tu programa.

La forma de lograr esto es la misma tanto en Linux como en Windows o Mac OS si ejecutas tu programa desde la consola de comandos:

Comando	Descripción
`./prg < ent.txt`	La entrada estándar de `prg` viene de `ent.txt`
`./prg > sal.txt`	La salida estándar de `prg` va a `sal.txt`
`./prg < ent.txt > sal.txt`	La entrada estándar de `prg` viene de `ent.txt` y la salida estándar va a `sal.txt`

Debes recordar que al vincular la entrada estándar a un archivo, éste debe existir y debe contener los datos de la entrada, pues de lo contrario ocurrirán errores durante la lectura de datos. Cuando la salida estándar se vincula a un archivo, éste se creará o sobreescribirá cada vez que ejecutes tu programa.

1. El toro celeste de Ishtar

`toro.c toro.cpp toro.java`

Después de que Gilgamesh despreciara a la diosa Ishtar, ésta en venganza envió al gran toro celeste a aterrorizar la ciudad de Uruk. El pequeño ejército de la ciudad fue tomado por sorpresa, pero decidió enfrentar a la bestia. Aunque el toro es muy fuerte, los hombres se dieron cuenta de que después de cada pelea, éste se debilita un poco: si el toro tiene fuerza inicial T y pelea contra un hombre con fuerza H, la fuerza del toro después de dicha pelea será $T-H$. El toro ganará la batalla si después de luchar contra los N hombres del ejército de Uruk, uno tras otro, todavía le queda fuerza. Escribe un programa que determine quién será el ganador de la batalla.

1.1. Entrada

Dos enteros T y N seguido de N enteros $H_1, H_2, \ldots, H_N$ que representan la fuerza de cada uno de los N hombres del ejército de Uruk. El toro luchará contra ellos en el orden de la entrada. Puedes suponer que $1 \leq T, N \leq 1\,000\,000$ y que $1 \leq H_i \leq 1\,000$.

1.2. Salida

Un entero G que sea la cantidad de hombres necesarios para dejar al toro sin fuerza, o el valor 0 si el toro ganó la batalla.

1.3. Ejemplo

Entrada	Salida
11 5 2 7 1 8 2	4

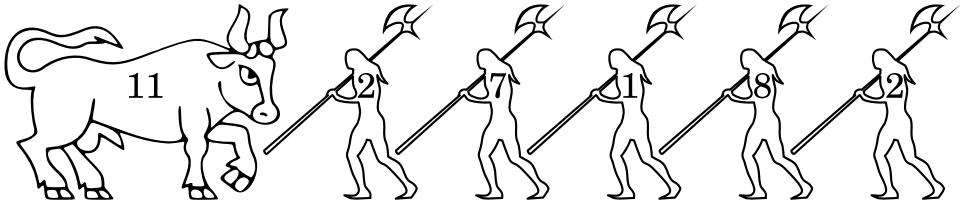

Figura 1: El toro tiene fuerza 11 y se requieren 4 hombres para derrotarlo.

2. La rama cristalina de Mashu

`rama.c rama.cpp rama.java`

En su búsqueda del secreto de la vida eterna, Gilgamesh tuvo que atravesar la peligrosa montaña Mashu. A la salida de ésta, Gilgamesh encontró unos misteriosos árboles con diamantes y cristales de antimonio en sus ramas. En el suelo se encontraba una rama caída y la tentación de llevársela completa era mucha, pero Gilgamesh creía que el antimonio era venenoso y que más de K de esos cristales serían mortales para él. Gilgamesh decide llevarse sólo un segmento de la rama, con la idea de llevarse la mayor cantidad de diamantes y a lo mucho K cristales. Escribe un programa que calcule cuál es la mayor cantidad de diamantes que se puede llevar bajo esta restricción.

2.1. Entrada

Dos enteros N y K, seguidos de una cadena R con N caracteres que denota la configuración de la rama. Una `D` en la posición R_i indica que el objeto i se trata de un diamante y una `A` que se trata de un cristal de antimonio. Puedes suponer que $1 \leq K \leq N \leq 100\,000$.

2.2. Salida

Un entero D que sea la cantidad máxima de diamantes que se puede llevar Gilgamesh.

2.3. Ejemplo

Entrada	Salida
10 2 DADADDDADA	5

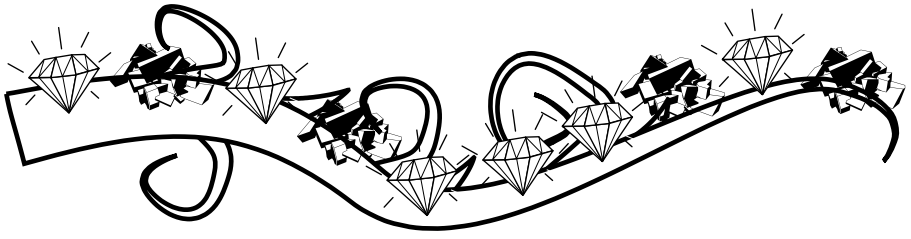

Figura 2: Se puede cortar un máximo de 5 diamantes con 2 cristales.

3. La barca cúbica de Shuripak

`barca.c barca.cpp barca.java`

Al encontrarse con el inmortal Utanapíshtim, Gilgamesh se sorprende de que no son tan distintos. Al preguntarle cómo es que ha logrado la vida eterna, Utanapíshtim le contesta que es una larga historia, pero que comenzó cuando el dios Enki descubrió el plan divino para inundar la corrupta ciudad Shuripak y ahogar a sus habitantes. Siendo Utanapíshtim una buena persona, Enki le advirtió que para salvar su vida y la de su familia debía construir una barca cúbica de lado L; la barca debía tener P pisos y cada piso debía dividirse en N partes a lo largo y M partes a lo ancho. Como la barca debía ser de madera, había que sellarla, por lo que Utanapíshtim barnizó con brea la superficie externa y con betún todas las superficies internas. Escribe un programa que calcule el área de las superficies que fueron barnizadas.

3.1. Entrada

Cuatro enteros L, P, N y M. Puedes suponer que $1 \leq L, P, N, M \leq 1000$.

3.2. Salida

Dos enteros A y B que sean las superficies que se barnizaron con brea y betún, respectivamente.

3.3. Ejemplo

Entrada	Salida
10 2 4 4	600 2000

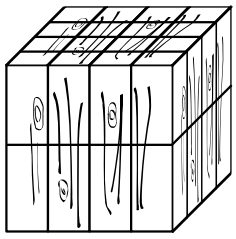

Figura 3: La barca tiene 2 pisos y cada uno se divide en 4×4 partes.