UNIVERSIDAD NACIONAL DE INGENIERÍA
Facultad de Ciencias / Ciencia de la Computación

Práctica de Laboratorio 03

Introducción a la Programación **CC102-AB**

---

1. (*2 puntos*) **Roman Digitis** /uva:344

Muchas personas están familiarizadas con los números romanos para un número relativamente pequeño. Los símbolos "I", "V", "X", "L", y "C" representan los valores decimales de 1, 5, 10, 50 y 100, respectivamente.

Para representar otros valores, estos símbolos se concatenan, con los símbolos de menor valor escrito más a la derecha.

Por ejemplo, el número 3 se representa como " III ", y el valor 73 se representa como " LXXIII".

Las excepciones a esta norma se producen para los números de unidades que tienen valores de 4 o 9, y para los valores de decenas de 40 o 90. Para estos casos, las representaciones de números romanos son "IV "(4), "IX"(9), "XL "(40), y "XC"(90).

Así que las representaciones de números romanos de: 24, 39, 44, 49, y 94 son "XXIV", "XXXIX","XLIV", "XLIX", y "XCIV ", respectivamente.

El prólogo de numerosos libros son numeradas con números romanos, a partir de "i" para la primera página del prólogo, y continuando en secuencia.

Asumiendo libros que tienen 100 o menos páginas de prefacio. ¿ Cuántos caracteres "i", "v", "x", "l ", y "c" son necesarios para enumerar las páginas del prólogo? Por ejemplo, en un prólogo de cinco páginas vamos a utilizar los números romanos "i", "ii", "iii", "iv", y "v", es decir, necesitamos 7 caracteres "i" y 2 caracteres "v".

La entrada consistirá en una secuencia de números enteros en el rango de 1 a 100, terminado por un cero. Para cada número entero, excepto el final cero, se determinará el número de diferentes tipos de caracteres necesarios para enumerar el prólogo con números romanos.

```
1
2
20
99
0
```

La Salida

Para cada número entero en la entrada, escriba una línea que contiene el entero de entrada y el número de caracteres de cada tipo requerido. Los ejemplos que se muestran a continuación ilustran un formato correcto.

```
1: 1 i, 0 v, 0 x, 0 l, 0 c
2: 3 i, 0 v, 0 x, 0 l, 0 c
20: 28 i, 10 v, 14 x, 0 l, 0 c
99: 140 i, 50 v, 150 x, 50 l, 10 c
```

---

2. (*2 puntos*) **Ugly Numbers** /uva:136

Los Ugly Numbers (Números Feos) son números cuyo único factores primos son 2, 3 o 5. La secuencia

```
1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, ...
```

muestra los primeros 11 números feos. Por convención, el uno está incluido.

Escribir un programa para encontrar e imprimir el número feo "1500'th".

Entrada y Salida

No hay ninguna entrada para este programa.

La salida debe consistir en una sola línea, como se muestra a continuación, donde <number>

debe ser sustituido por el número calculado.

Ejemplo de salida

```
The 1500'th ugly number is <number>.
```

---

3. (*2 puntos*) **Mirror, Mirror** /uva:466
Un patrón de cuadrados de celdas claras y oscuras se muestra en su estado original y un estado transformado. Escriba un programa que reconozca la transformación mínima que se ha aplicado al modelo original dado la siguiente lista de posibles transformaciones:

1. **90 grados de rotación:**
   El patrón fue girado a la derecha 90 grados.
2. **180 grados de rotación:**
   El patrón fue girado a la derecha 180 grados.
3. **270 grados de rotación:**
   El patrón fue girado a la derecha 270 grados.
4. **Vertical de reflexión:**
   El patrón se refleja a través de un espejo horizontal, colocado sobre el patrón.
5. **Combinación:**
   El modelo fue sometido a una reflexión vertical seguida de una de las rotaciones.
6. **Conservación:**
   El modelo original se conserva (el nuevo modelo es idéntico al original).
7. **Incorrecto:**
   El nuevo modelo no se obtuvo a través de cualquiera de estos treansformaciones.

La entrada consistirá en un número desconocido de conjuntos de datos patrón.
Cada conjunto de datos contará con un número entero "m" en la primera línea, indicando las dimensiones del cuadrado que contiene el patrón (el tamaño que van desde 1 a 10). Las "m" líneas siguientes contienen los patrones originales y nuevos (transformado) en un formato de lado a lado, separados por un espacio.
Las casillas con luz serán indicados por un punto, mientras casillas negras estarán representadas con una X.
La salida será una frase que describe la relación que el nuevo modelo lleva a la original.
Cada frase se iniciará con un número de identificación del patrón (a partir de 1) y al final indicando la relación que representa la cantidad mínima de trabajo necesario para obtener el nuevo patrón a partir del original.
A los efectos de evaluar la cantidad de trabajo necesario, las rotaciones son considerados menos trabajo que las reflexiones y rotaciones más pequeñas son menos trabajo que las grandes.
Por supuesto, la "preservación" implica ningún trabajo en absoluto.
Tenga en cuenta que sólo las posibilidades mencionadas deben ser considerados - no existe "rotación de 360 grados" para este problema, ni tampoco hay una "reflexión horizontal".
Además, recuerde que cuando una simple rotación o reflexión no es suficiente, su programa deberá considerar las rotaciones después de una reflexión vertical.
A pesar de que una transformación combinada podría producir el nuevo patrón, igual que una transformacion individual, la transformación simple es la que debe mostrarse en la salida (transformación mínima).
El resultado debe ser una oración completa, que termina con un punto.
Observe la salida del ejemplo siguiente para el formato exacto.

Por ejemplo para la entrada:

```
5
X...X  ....X
.X...  ...X.
...X.  .X...
..X.X  ..X..
....X XX..X
6
....XX X....X
...X..  X.X...
XX..X.  .X..X.
..X...  ...X.X
...X..  ..X...
..X..X  ..X...
2
X.  X.
.X  .X
4
..X.  ...X
XX..  ....
....  XX..
...X  ..X.
5
X....  .X...
.X...  ..X..
.X...  ..X..
...X.  ....X
....X X....
4
.X..  ..X.
.X.X X...
....  ..XX
..X.  ....
2
..  XX
XX  ..
```

Obtendríamos:

```
Pattern 1 was rotated 90 degrees.
Pattern 2 was rotated 270 degrees.
Pattern 3 was preserved.
Pattern 4 was reflected  vertically .
Pattern 5 was improperly transformed.
Pattern 6 was reflected  vertically  and rotated 270 degrees.
Pattern 7 was rotated 180 degrees.
```

---

4. (*2 puntos*) **Encoder and Decoder** /uva:444

El encargado del departamento de informática de la Agencia Internacional de Espionaje, le pide escribir un programa que permita a un espía codificar y decodificar sus mensajes.
Puede asumir que el mensaje de un espía tiene como maximo 80 caracteres, e incluye todas las letras mayúsculas y minúsculas del alfabeto, más el espacio, y cualquiera de los siguientes caracteres: “!”,“,”,“.”,“:”,“;”,“?” La siguiente es una tabla ASCII de los caracteres válidos en un mensaje:

”A”	65	”a”	97	”␣”	32
”B”	66	”b”	98	”!”	33
.		.		”,”	44
.		.		”.”	46
.		.		”:”	58
”Y”	89	”y”	121	”;”	59
”Z”	90	”z”	122	”?”	63

El algoritmo que se debe utilizar para codificar los mensajes es tomar el valor ASCII de cada carácter del mensaje, comenzando del final y terminando con el primer carácter del mensaje. Se debe ir agregando al mensaje codificado este valor ASCII escrito en orden inverso.
Por ejemplo, si el valor ASCII es “123”, el mensaje codificado debe contener la cadena “321”. No debe haber espacios que separan los números en el mensaje codificado.
La entrada consiste en una o más líneas con un mensaje normal (no codificados) o codificado.
La salida debe tener el mismo número de líneas, con el correspondiente mensaje codificado o decodificado respectivamente.
Por ejemplo para la entrada:

```
abc
798999
Have a Nice Day !
```

La salida sería:

```
998979
cba
33231217986231019950187237923101811792 7
```

---

5. (*2 puntos*) **The Skyline Problem** /uva:105

Con la llegada de la alta velocidad de los gráficos, estaciones de trabajo CAD (diseño asistido por computador) y otras áreas (CAM, diseño VLSI) han hecho un uso cada vez más eficaz de los computadores. Uno de los problemas con las imágenes de dibujo es la eliminación de líneas ocultas - líneas ocultas por otras partes de un dibujo.

Debes diseñar un programa para ayudar a un arquitecto en la elaboración de la silueta de una ciudad, dada la ubicación de los edificios de la ciudad.

Para hacer manejable el problema, todos los edificios son de forma rectangular y comparten un fondo común (la ciudad se construyen en el mismo plano).

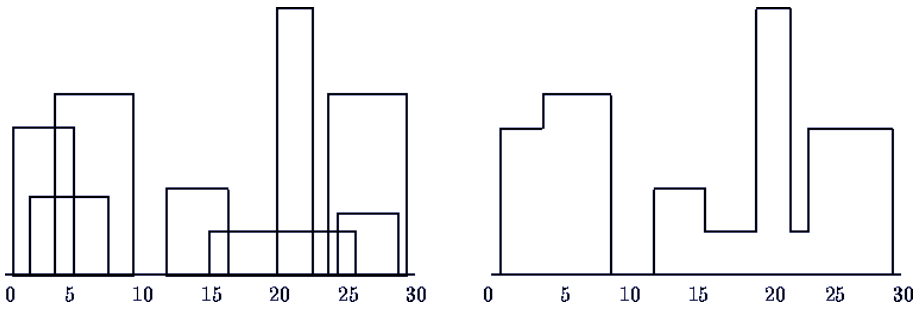
La ciudad es vista en dos dimensiones.

Un edificio se especifica por un triple (tres numeros) "(Li,hi,Ri)", donde "Li" y "Ri" son las coordenadas izquierda y derecha, respectivamente, y "Hi" es la altura del edificio.

En el siguiente diagrama a la izquierda se muestran los edificios con triples (1,11,5), (2,6,7), (3,13,9), (12,7,16), (14,3,25), (19,18,22), (23,13,29), (24,4,28)

El horizonte, que se muestra a la derecha y está representado por la secuencia:

(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)



La entrada es una secuencia de triples (edificio).

Todas las coordenadas de los edificios son enteros positivos menores que 10,000 y habrá por lo menos uno y un máximo de 5.000 edificios.

Cada edificio se encuentra en una línea (triple).

Todos los números enteros en un triple están separados por uno o más espacios.

Los triples, estarán ordenados por "Li", la coordenada-x izquierda del edificio, por lo que el edificio con la más pequeña coordenada-x a la izquierda es el primero en la entrada.

La salida debe consistir en el vector que describe el horizonte como se muestra en el ejemplo anterior.

En el vector horizonte (v1,v2,v3,...vn), el "vi" tal que "i" es un número par representa la altura.

El "vi" tal que "i" es un número impar representa la coordenada-x.

El vector horizonte debe representar un camino continuo, partiendo de la mínima coordenada x y viajar horizontalmente y verticalmente sobre todas las líneas que definen el horizonte. Así, la última entrada en el vector horizonte será un 0. Las coordenadas deben estar separadas por un espacio en blanco.

Ejemplo de entrada

```
1  11  5
2  6  7
3  13  9
12  7  16
14  3  25
```

```
19  18  22
23  13  29
24  4  28
```

Ejemplo de salida

```
1  11  3  13  9  0  12  7  16  3  19  18  22  3  23  13  29  0
```

---

6. (*2 puntos*) **Student Grants** /uva:144

El Gobierno de Impecunia ha decidido a disuadir a los estudiantes terciarios, haciendo de los pagos de becas terciarias un proceso largo y lento.

Cada estudiante recibe una tarjeta de identificación de estudiante que tiene una banda magnética codificada en la parte posterior que registra el pago de la beca de estudios. Esto se establece inicialmente en cero. La subvención se ha fijado en $40 po r año y se paga al estudiante el día de trabajo más cercano a su cumpleaños. (La sociedad Impecunian es todavía algo medieval y solamente hombres continuan con la educación terciaria.) Así, en un día cualquiera de trabajo, hasta 25 alumnos aparecerán en la oficina más cercana del Departamento de Subvenciones para obtener su concesión.

La subvención se paga por un cajero automático. El cajero automático fue construido en los Talleres de Estado y está diseñado para ser difícil de robar. Consiste en una bóveda interior, donde se mantiene un gran stock de monedas de $ 1 y un almacén de salida de estas monedas que se dispensan. Para limitar las posibles pérdidas, sólo se mueven monedas de la bóveda al almacen de salida cuando esta queda vacía.

Cuando la máquina se enciende por la mañana, con un almacén de salida de vacío, de inmediato se mueve una moneda al almacén de salida. Cuando eso se ha dispensado entonces se moverá dos monedas, luego 3, y así sucesivamente hasta llegar a un límite preestablecido k. A continuación, se regresa de nuevo a 1, luego 2 y así sucesivamente.

Los estudiantes forman una cola en la máquina y, esperan su turno, cada estudiante inserta su tarjeta. La máquina dispensa lo que tiene en su almacén de salida y actualiza la cantidad pagada a ese estudiante escribiendo el nuevo total en la tarjeta. Si el estudiante no ha recibido su beca completa, retira su tarjeta y vuelve a la cola pero al final. Si la cantidad en el almacen de salida, es más de lo que, al estudiante le correspondeía recibir para completar sus $ 40, la máquina sólo paga lo suficiente como para hacer que el total de hasta $ 40. Dado que este hecho se registra en la tarjeta, no tiene sentido para el estudiante volver a realizar cola y se va.

La cantidad restante en el almacén está entonces disponible para el estudiante que viene.

Escriba un programa que lea en los valores de N (el número de estudiantes, entre 1 y 25) y k (el límite para esa máquina, entre 1 y 40), calcular el orden en que los estudiantes salen de la cola. La entrada consistirá en una serie de líneas, cada una con un valor de N y K, enteros.

La lista se dará por terminado por dos ceros (0 0). La salida consistirá en una línea para cada línea de entrada y contendrá la lista de los estudiantes en el orden en que salen de la cola. Los estudiantes están ordenados según su posición en la cola al inicio del día. Todos los números deben justificarse a la derecha en un campo de ancho 3.

Para la entrada:

```
5  3
0  0
```

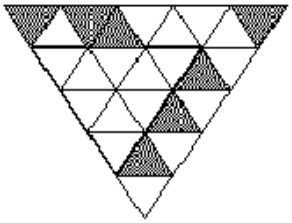
Se obtendrá:

```
  1   3   5   2   4
```

---

7. (*2 puntos*) **Triangles** /uva:585

Con una estructura de triángulo con campos en blanco y negro en el interior debe hallar el área más grande del triángulo de campos de blanco, como se muestra en la siguiente figura:



La entrada contiene varias descripciones de triángulos. La primera línea de cada descripción contiene un entero “n” (entre 1 y 100), lo que da la altura del triángulo. Las “n” líneas siguientes contienen caracteres del conjunto espacio, #, - que representan las filas del triángulo, donde # es un negro y ’-’ un campo blanco. Los espacios se utilizan sólo para mantener la forma de triángulo.
(Compárese con la entrada de la muestra. El primer caso de prueba corresponde a la figura.)
Para cada triángulo, el número de los caracteres ’#’ y ’-’ por línea es impar y disminuye de “2n - 1” a “1”.
La entrada termina con una descripción de partida con n = 0.
Para cada triángulo se mostrará la mayor área de triángulo que se pueda formar con los campos blancos, tal como se muestra en el ejemplo. Cada caracter representa una unidad de área.
Tenga en cuenta que el mayor triángulo puede tener su punto en la parte superior, como en el segundo caso de la entrada de la muestra. Una línea en blanco separa cada caso.
Ejemplo de entrada

```
5
\#-\#\#----\#
 -----\#-
  ---\#-
   -\#-
    -
4
\#-\#-\#--
 \#---\#
  \#\#-
   -
0
```

Ejemplo de salida

```
Triangle \#1
The largest triangle area is 9.

Triangle \#2
The largest triangle area is 4.
```

---

8. (*2 puntos*) **Triangle Wave** /uva:488

El problema consiste en generar una forma de onda triangular, de acuerdo a un par específico de amplitud y frecuencia.
La entrada comienza con un solo número entero positivo en una línea, que indica el número de los casos. Esta línea es seguida por una línea en blanco, y también hay una línea en blanco entre dos casos consecutivas.
Cada caso contendrá dos enteros, cada uno en una línea separada. El primer entero es la amplitud, el segundo entero es la frecuencia.
Los resultados de dos casos consecutivos se separan por una línea en blanco.
El número total de formas de onda es igual a la frecuencia, y la altura horizontal de cada onda es igual a la amplitud. La amplitud nunca será superior a nueve.
La propia forma de onda se debe llenar con números enteros en cada línea que indican la altura

de esa línea.
NOTA: Hay una línea en blanco después de cada forma de onda particular, excepto el último.

Por ejemplo para la entrada:

```
2

4
2

7
1
```

Se produciría:

```
1
22
333
4444
333
22
1

1
22
333
4444
333
22
1

1
22
333
4444
55555
666666
7777777
666666
55555
4444
333
22
1
```

---

9. (*4 puntos*) **Chess** /uva:278

Casi todo el mundo conoce el problema de colocar ocho reinas en un tablero de ajedrez de 8 x 8, de tal manera, que una reina no puede tomar otra reina.
Jan Timman (un famoso ajedrecista holandés) quiere saber el número máximo de piezas de ajedrez de un tipo que se pueden poner sobre una tablero de " m x n ", de tal manera que ninguna pieza puede tomar a la otra. Debido a que es bastante difícil encontrar una solución a mano, le pide su ayuda para resolver el problema.
Él no necesita saber la respuesta para todas las piezas.
Él sólo quiere saber cuantas Torres, Caballos, Reinas o Reyes se pueden colocar en un tablero, de tal manera que una pieza no puede tomar a ninguna otra.

La primera línea de **entrada** contiene el número de casos. Cada caso aparece en una línea y se compone de un caracter indicando el tipo de pieza (r=Torre,k=Caballo,Q=Reina y K=Rey), seguido por el entero "m" y "n", conteniendo el número de filas y el número de columnas respectivamente, tal como se muestra:

```
2
r 6 7
k 8 8
```

La salida mostrará para cada caso el número máximo de piezas que se pueden poner en el tablero indicado.

```
6
32
```

Nota: El cuadro inferior izquierdo es 1, 1.

---

10. (*4 puntos*) **Bit Maps** /uva:183

El mapa de bits es una estructura de datos que se plantea en muchas áreas de la informática. En el área de gráficos, por ejemplo, un mapa de bits puede representar una imagen con un 1 representa un píxel negro y un 0 representa un píxel blanco. Considere las siguientes dos maneras de representar un mapa de bits rectangular.

En el primero, es simplemente representado como una matriz bidimensional de 1s y 0s.

El segundo se basa en una técnica de descomposición. En primer lugar, se considera el mapa de bits completo. Si todos los bits dentro de el son 1, 1 es una salida. Si todos los bits dentro de el son 0, 0 es la salida. De lo contrario, una D es de salida, el mapa de bits se divide en cuatro partes (como se describe más adelante), y cada uno de ellos es procesado en la misma forma que el mapa de bits original.

Los cuartos se procesan en el orden siguiente: superior izquierda, superior derecha, inferior izquierda, e inferior derecha.

Cuando un mapa de bits se divide y tiene un número par de filas y un número par de columnas, todos los cuartos tienen las mismas dimensiones. Cuando el número de columnas es impar, los cuartos izquierdos tienen una columna mas que los de la derecha. Cuando el número de filas es impar los cuartos superiores tienen una fila más que los de la parte inferior.

Escriba un programa que lea en mapas de bits de cualquier forma y transformar a la otra forma.

La entrada consistirá en una serie de mapas de bits. Cada mapa de bits se inicia con una línea que indica su formato ("B" o "D") y sus dimensiones (filas y columnas), tres argumentos separados por al menos un espacio. Ninguna dimensión será superior a 200. Siguiendo a esta línea estará una o más líneas conteniendo la secuencia de " 1"," 0" y " D", que representan el mapa de bits, sin espacios intermedios. Cada línea (excepto la última, que puede ser más corta) contendrá 50 caracteres.

Un mapa de bits de tipo "B" se escribe de izquierda a derecha y de arriba abajo. La entrada de datos termina con una línea conteniendo un #. Por ejemplo para la entrada:

```
B 3   4
001000011011
D   2     3
DD10111
#
```

La salida de cada uno de mapa de bits se inicia en una nueva línea y será en el mismo formato que el de entrada. El ancho y el alto se justificaran a la derecha en campos de ancho 4, tal como se muestra.

```
D     3     4
D0D1001D101
B     2     3
101111
```
