COMPUTACIÓN 1
Instituto de Computación

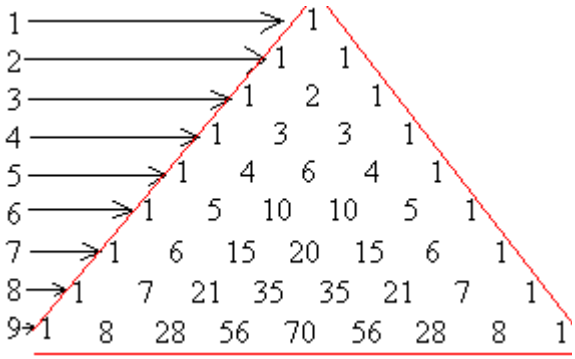
2do Parcial - 24 de noviembre de 2011- 2a parte

- No se podrá utilizar ningún tipo de material (apuntes, libro, calculadora, etc). Apague su teléfono celular.
- **Sólo** se contestarán preguntas sobre interpretación de la letra hasta 20 minutos antes de la finalización del mismo.
- Escriba las hojas de un sólo lado.
- Las partes no legibles del examen se considerarán no escritas.
- En la primer hoja a entregar ponga con letra clara, en el ángulo superior derecho, su nombre, número de cédula de identidad y cantidad de hojas -en ese orden-; las demás hojas es suficiente con nombre, número de cédula y número de página.

Problema 1 (6 y 4 puntos, respectivamente)

El triángulo de Pascal es una representación de los coeficientes binomiales ordenados en forma de triángulo.

El triángulo se construye de la siguiente manera. En el primer nivel se coloca solamente el número 1 centrado en la parte superior. Cada nivel se construye con un número 1 al principio y un número 1 al final. Los elementos intermedios de cada nivel se calculan sumando el par de números del nivel superior que se encuentran contiguos a la posición que se quiere calcular. Por ejemplo, en el nivel 3 el valor 2 se calcula como la suma de los dos números 1s del nivel 2. Notar que en el nivel 2 no hay elementos intermedios.



Se pide:
Parte a) Implemente una función **iterativa** en ***Matlab*** *CalcularSiguiente* que reciba un vector de largo **L** con los elementos de un nivel del triángulo de Pascal y retorne un vector con los elementos del siguiente nivel del triángulo.

Parte b) Implemente una función **iterativa** en ***Matlab*** *CalcularNivel* que reciba como parámetro el número de nivel del triángulo de Pascal que se quiere calcular y devuelva el vector con los elementos de dicho nivel del triángulo.

Notas: - Para resolver la parte b) se puede usar la parte a).
- No se puede usar suma de vectores, ni se puede usar el comando sum o cualquier otro comando de Matlab que ayude a resolver trivialmente el problema.

Problema 2 (8 puntos)

La función de Ackermann ($\mathbb{N}\times\mathbb{N}\rightarrow\mathbb{N}$) tiene la siguiente expresión:

$$A(m,n)=\begin{cases} n+1 \quad si \quad m=0 \\ A(m-1,1) \quad si \quad n=0 \\ A(m-1,A(m,n-1)) \quad si \quad n>0\, y\, m>0 \end{cases}$$

Se pide elaborar una función **recursiva** de ***Matlab*** *Ackermann* la cual recibe los naturales **m** y **n**, y retorne el valor de la función de Ackermann.

Problema 3 (12 y 5 puntos, respectivamente)

Se desea implementar una función que calcule la suma de dos números en una base determinada. Dicha función recibirá un número entero positivo **N** que indica la base y dos vectores de enteros **a** y **b**. Cada elemento *x* de los vectores se corresponde con un dígito del número que representa y cumple que $0 \leq x < N$.
Como parámetro de salida se debe retornar el vector suma que contiene los elementos de $a+b$ usando la misma representación que los parámetros de entrada (vector de enteros en base *N*).

Parte a) Implemente la función **recursiva** en ***Matlab*** *sumaCarry* que recibe como parámetros de entrada dos vectores de enteros, **a** y **b**, y un número natural **N, y** retorne como parámetros de salida el vector suma y el acarreo generado. Asuma que los vectores de entrada tienen el mismo largo. El vector suma tiene que tener el mismo largo que los parámetros de entrada.

Parte b) Implemente la función en ***Matlab*** *sumaBase* que reciba como parámetros de entrada dos vectores de enteros, **a** y **b**, y un número natural de **N, y** retorne como único parámetro de salida el vector suma. No asuma que los vectores de entrada tienen el mismo largo. El vector suma que se retorne puede tener un largo mayor que los parámetros de entrada. Utilice la función implementada en la parte *a*) para calcular la suma.

Ejemplos:

```
>> [suma,carry] = sumaCarry( [0, 14, 3], [15, 12, 4], 16 )
suma =
        0    10    7
carry =
        1

>> suma = sumaBase( [14, 3], [15, 12, 4], 16)
suma =
        1    0    10    7

>> suma = sumaBase( [15, 12, 4], [14, 3], 16)
suma =
        1    0    10    7
```

Nota: - No se puede utilizar la suma de vectores, ni se puede usar el comando sum o cualquier otro comando de Matlab que ayude a resolver trivialmente el problema.
- Se puede utilizar la suma de enteros.

Problema 4 (5, 5 y 5 puntos, respectivamente)

Dada una matriz dispersa en formato elemental que no posee coeficientes negativos y un valor **j** que indica una columna, se pide:

Parte a) Implemente una función **iterativa** en ***Matlab*** *MaxCol* que calcule el máximo elemento correspondiente a la columna **j**.

Parte b) Implemente una función **recursiva** en ***Matlab*** *MaxColRec* que calcule el máximo elemento correspondiente a la columna **j**.

Dada una matriz dispersa en formato elemental que no posee coeficientes negativos y un valor **m** que indica la cantidad de columnas que posee la matriz, se pide:

Parte c) Implemente una función **iterativa** en ***Matlab*** *MaxTodasCol* que reciba el indicador m y la matriz dispersa, y retorne un vector con el máximo valor para cada columna de la matriz. Para la resolución de este ejercicio no se puede utilizar las funciones de la parte *a)* o *b).*

Nota: - No asuma ningún orden en los elementos de la matriz.
- Puede asumir que todos los parámetros de las funciones son correctos.
- No se puede operar con matrices en formato completo.