COMPUTACIÓN 1
Instituto de Computación

Examen - 28 de diciembre de 2020
Parte 2: Iteración – 42 puntos

- **Duración de esta parte de la prueba: 75 minutos.**
- **Duración total incluyendo descansos: 200 min (45' – 10' – 75' – 10' - 60') = 3h:20min**
- **Puntaje total de la prueba: 100 puntos.**
- 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.

Para la resolución de los diferentes ejercicios **solamente** podrá utilizar las siguientes funciones brindadas por **Octave**:

- `length() y size()`
- `mod() y rem()`
- `floor(), ceil() y round()`
- `zeros() y ones()`

Notas: - En todos los ejercicios se deben usar las estructuras de control adecuadas para cada caso. Por ejemplo: se controlará el uso correcto de for y while.
- No se deben realizar más iteraciones que las necesarias para resolver cada uno de los problemas

Problema 4	14 ptos	

Implementar en *Octave* la función **iterativa** *primeraFilaQueSumaX* que, dada una matriz **M** que tiene números y un número ***X***, devuelva el indice de la primera fila de **M** cuya suma sea igual a ***X***. En caso de que no haya ninguna fila que sume ***X***, la función debe devolver -1. La matriz **M** contiene números positivos y negativos.

Problema 5	14 ptos	

Sea un vector ***posiciones*** que contiene números enteros mayores o iguales a 1 y menores o iguales al largo del vector. Los valores almacenados en el vector ***posiciones*** permiten realizar una recorrida del propio vector saltando a la posición indicada por cada elemento. Por ejemplo, si el vector es [3, 4, 2, 4, 2] y la posición inicial es **1**, se debe consultar el valor en ***posiciones(1)***. Como este valor es 3, el siguiente valor a consultar es el que está en ***posiciones(3)***, y así sucesivamente. Nótese que, como se observa en el ejemplo anterior, el vector puede tener elementos repetidos (como el 2 y el 4) y pueden existir elementos que no aparezcan en el vector (como el 1 y el 5).

Implementar en *Octave* la función ***iterativa*** *saltaHastaRepetir* que, reciba una posición inicial y un vector ***posiciones*** con las características descritas en el párrafo anterior, y devuelva la mínima cantidad de saltos que deben realizarse desde la posición inicial para caer en una posición ya visitada.

```
saltaHastaRepetir(6,[3, 1, 4, 2, 3, 6]) devuelve 1
saltaHastaRepetir(1,[3, 1, 4, 2, 3, 6]) devuelve 4
saltaHastaRepetir(5,[3, 1, 4, 2, 3, 6]) devuelve 5
```

Sugerencia: Utilice un vector auxiliar para recordar las posiciones del vector original que ya fueron visitadas.

Problema 6	14 ptos	

Implementar en *Octave* la función **iterativa** *multiplicacionElementoaElementoFiltrada,* que dadas dos matrices **A** y **B** completas y una matriz **S** dispersa en **formato elemental**, devuelva una matriz dispersa en **formato elemental D** que contenga el producto "elemento a elemento" de **A** y **B** en las posiciones de **S** que son distintas de 0, siempre y cuando dicho producto no dé 0. Es decir **D(i,j)=A(i,j)*B(i,j)** si y solo si existe un elemento distinto de cero en la matriz dispersa **S** en la posición (i,j) (fila i y columna j) y el producto **A(i,j)*B(i,j)** es distinto de 0. **A** y **B** tienen exactamente las misma dimensiones. Los valores de las filas y las columnas de los elementos no ceros de **S** están dentro de las dimensiones de **A** y **B.**