**Pontificia Universidad Católica Madre y Maestra**
**Facultad de Ciencias de La Ingeniería** **Primer Parcial**
**Departamento de Ingeniería de Sistemas y Computación**
**Algoritmos Fundamentales-ISC-201** Matrícula:_____

**Resuelva los siguientes problemas, cada uno vale 25%.**

1. La persistencia de un entero es el número de veces que hay que multiplicar sus dígitos hasta alcanzar un solo dígito. Así la persistencia de 715 se calcula multiplicando cada dígito del número entre sí. Es decir, 7*1*5 = 35. Como 35 no es de un dígito se continúa multiplicando los dígitos de 35 entre sí. Es decir, 3*5 = 15. De igual forma 15 no es de un dígito, se continúa multiplicando los dígitos de 15 entre sí. Es decir, 1*5 = 5 como 5 es de un dígito no se multiplica más y se cuenta las veces que fue necesario multiplicar los dígitos de cada valor resultante, que en este caso fueron 3. Por ende la persistencia de 715 es 3. Realice una función que dado un entero determine su persistencia.

```
int persistencia(int numero)
{
   int persist = 0, nuevo_numero;

   // Mientras el número tenga más de un dígito...
   while( numero > 9 )
   {
      // Para poder iniciar serie multiplicativa
      nuevo_numero = 1;
      while (numero != 0 )
      {
         // Tomo el último dígito y
         // lo multiplico por el multiplicativo
         nuevo_numero *= persist % 10;
         // Eliminando el último dígito
         numero /= 10;
      }
      // Cuento cuantas veces multiplico
      persist ++;

      // Asigno a número el nuevo
      // número resultado de la multipicaciones...
      numero = nuevo_numero;
   }

   return persist;
}
```

2. La traza de una matriz cuadrada es la suma de los elementos de la diagonal superior. Se dice que una matriz está balanceada cuando la traza es mayor que el promedio de los elementos no incluidos en la diagonal superior. Realice la función **int balanceada(float m[ ][COL], int n)** que retorne 1 si la matriz **m** está balanceada y 0 si no lo está.
**Ejemplo: Sea m =**

| | | |
|---|---|---|
| 24 | 12 | 16 |
| 32 | 3 | 23 |
| 33 | 22 | 99 |

La traza de **m** es 24 + 3 + 99 = **126**.
El promedio de los números no incluidos en la diagonal superior:
12 + 16 + 23 + 32 + 22 + 33 = 138 / 6 = **23.**
Como la traza (**126**) es mayor que el promedio de los números no incluidos en la diagonal superior **(23**), entonces esta matriz está balanceada.

**SOLUCIÓN A:**

```
int balanceada (int m[ ][COL], int n)
{
   int fila, columna, traza = 0, suma = 0, cantidad = 0;

   for (fila = 0; fila < n; fila ++ )
      for (columna = 0; columna < n; columna ++)
         // Diagonal superior...
         if ( fila == columna )
            traza += m[fila][columna];
         else
         {
            suma += m[fila][columna];
            cantidad ++;
         }

   if (traza > (suma / cantidad) )
      return 1;

   return 0;
}
```

**SOLUCIÓN B**:

```
int traza (int m[ ][COL], int n)
{
   int fila, columna, suma = 0;

   for (fila = 0; fila < n; fila ++ )
      for (columna = 0; columna < n; columna ++)
         // Diagonal superior...
         if ( fila == columna )
            suma += m[fila][columna];

   return suma;
}


int prom_no_diag_sup(int m[ ][COL], int n)
{
   int fila, columna, suma = 0, cantidad = 0;

   for (fila = 0; fila < n; fila ++ )
      for (columna = 0; columna < n; columna ++)
         // No es Diagonal superior...
         if ( fila != columna )
         {
            suma += m[fila][columna];
            cantidad ++;
         }

   return suma / cantidad;
}


int balanceada(int m[ ][COL], int n)
{
   return traza(m,n) > prom_no_diag_sup(m,n);
}
```

3. Dado la cadena de caracteres **s,** se desea saber cuántas vocales hay en la misma. Realice la función **int cantidadvocales(char s[ ])**, que retorne el total de vocales contenidas en **s.**

```
int cantidadvocales(char s[ ])
{
   int ind, total = 0;
   char caract_eval;

   for (ind = 0; s[ind]; ind ++ )
   {
      // Llevo a minúsculas
      caract_eval = tolower(s[ind]);
      if ( caract_eval == 'a' || caract_eval == 'e' ||
           caract_eval == 'i' || caract_eval == 'o' ||
           caract_eval == 'u' )
         total ++;
   }

   return total;
}
```

4. Dados tres arreglos de números de dimensión **n** se desea una función que permita obtener el cuadrado de la diferencia de los dos primeros en el tercer arreglo.

   Por ejemplo: Si se tienen 3 arreglos A1, A2, A3 de dimensión 5 y con los siguientes elementos:

   A1

| 12.34 | 23.56 | 18.45 | 20.32 | 10.23 |
|---|---|---|---|---|

   A2

| 10.34 | 18.45 | 32.12 | 25.23 | 18.45 |
|---|---|---|---|---|

   Luego de haber llamado la función en cuestión el arreglo A3 tendría:

| 4.0 | 26.1121 | 186.8689 | 24.1081 | 67.5684 |
|---|---|---|---|---|

```
void opera_arr(int a1[ ], int a2[ ], int a3[ ], int n)
{
   int ind;

   for (ind = 0; ind < n; ind ++ )
      a3[ind] = pow(a1[ind] - a2[ind], 2.0);

   return;
}
```