Práctico 1 - Introducción

Ejercicio 1 - Función módulo (MOD)

La función módulo, más conocida por su abreviación MOD, calcula el residuo de una operación de división entera, de modo que 7 MOD 3 = 1. Generalizando, $x\ MOD\ y$ lo podemos escribir como $x - k \cdot y$, donde k es el entero inferior más cercano del cociente x/y.

a) Escribir un programa que calcule el MOD de dos números y lo imprima en pantalla. Estos números serán fijados como constantes en el programa.

b) Escribir una función `mod` que tome dos enteros como parámetros y devuelva el módulo de ambos como resultado entero. Probarlo con un programa ejecutable como el que se muestra debajo (recuerde que la función a implementar tiene que estar definida antes del comienzo de `main()`).

```
#include <stdio.h>

int main()
{
  int m;
  m = mod(4,5);
  /* %d se rellena con el valor de m al imprimirse la cadena */
  printf("mod(17,5)=%d\n", m);
  return m;
}
```

Ejercicio 2 - Simulador de dados

Realizar un programa que simule la tirada de un dado, devolviendo un valor aleatorio entre 1 y 6 en cada invocación. Se sugiere investigar respecto a la función `rand()`[1] contenida en la librería estándar.

[1] *The C Programming Language* Kernighan & Ritchie - Apéndice B5

Ejercicio 3 - La Gaussiana

La función normal utilizada en aplicaciones estadísticas tiene la siguiente expresión:

$$y = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

a) ¿Cuántos parámetros de entrada tiene esta función?

b) Realizar un programa que implemente la función normal, devolviendo el valor en punto flotante (`float`) de y resultante para cualquier combinación de sus parámetros de entrada.

c) Escriba un `main` que evalúe la función para los valores $\mu = 1$, $\sigma = 4$, $x = 2$ y lo imprima en pantalla. Verifique su resultado con una calculadora.

NOTA: *Se sugiere investigar en la bibliografía sugerida la librería <math.h>; la misma contiene las funciones* ***`sqrt`*** *y* ***`exp`****, y la constante numérica* ***`MATH_PI`****, que pueden resultar de gran utilidad.*
Además, para poder compilar exitosamente programas con estas funciones, es necesario agregar la opción ***`-lm`*** *al final de la linea de comandos de compilación, ejemplo:*

```
$cc gauss.c -lm -o gauss
```

Ejercicio 4 - Verificando condiciones

Realizar un programa que calcule e imprima la raíz cuadrada y el inverso de un número fijo. Antes de calcular la raíz se debe verificar que el número no sea negativo y antes de calcular el inverso se debe verificar que el número no sea nulo. De no cumplirse estas condiciones el programa debe alertar al usuario de estos problemas mediante un texto en pantalla.

Ejercicio 5 - Argumentos desde la linea de comandos

Es posible capturar los argumentos desde la línea de comandos si la función `main` es declarada de modo que tome dos parámetros `argc` y `argv` (el tipo de datos `char **` de `argv` sera visto mas adelante):

```
#include <stdio.h>

int main(char argc, char **argv)
{
  int n;
  printf("Número de argumentos: %d\n",argc);
  n = 0;
  while (n < argc) {
    printf("Argumento %d vale %s\n", n, argv[n]);
    n++;
  }
  return n;
}
```

a) Compile y ejecute el código anterior para distintos parámetros cualesquiera.

En los clásicos descuentos del impuesto al valor agregado (IVA), hoy en día 22 %, mucha gante comete el error de quitar el 22 % del precio de plaza, olvidando que este es el resultado de agregarle el 22 % al precio original sin la carga impositiva. De modo que $Precio\_de\_plaza = 1,22 \cdot Precio\_original$.

b) Realizar un programa que reciba el precio de plaza y retorne el precio original. Modificar el programa anterior de forma que el valor del IVA sea ingresado por el usuario, esto brindará robustez frente a cambios en la política impositiva.

Ejercicio 6 - Dígitos decimales de un número entero

Escriba un programa que imprima uno a uno los dígitos decimales de un número entero dado. Se sugiere utilizar la función `mod` vista en ejercicios anteriores.

Ejercicio 7 - Dígito verificador

La Dirección Nacional de Identificación Civil (DNIC) nos adjudica en la cédula de identidad un número único para cada habitante de nuestro país. Dicho número contiene un dígito verificador generado automáticamente. Dados los 7 dígitos de un número de cédula $c_0c_1c_2c_3c_4c_5c_6$, siendo c_0 los millones y c_6 las unidades, el algoritmo para calcular el dígito verificador v es el siguiente:

1. $s = 2c_0 + 9c_1 + 8c_2 + 7c_3 + 6c_4 + 3c_5 + 4c_6$
2. r es el menor múltiplo de 10 mayor a s
3. $v = r - s$

A modo de ejemplo, para el número de cédula 4213264 se tiene $s = 101$, luego $r = 110$, y finalmente $v = 9$.

a) Escriba una función `verif` que tome como argumentos los 7 dígitos de la cédula como un arreglo de enteros y devuelva el dígito de verificación como otro entero. Escriba un `main` para probarla con una o más cédulas conocidas fijas en el código (por ejemplo la suya).

b) Escriba un programa que reciba el número de cédula (sin verificación) desde la línea de comandos y lo muestre luego con su dígito verificador al final (tal como se muestra en la cédula, separado por un guión). Para convertir la cadena de texto ingresada a un número entero de 7 dígitos, utilice la función de la biblioteca estándar `atoi`, por ejemplo

```
int main(char argc, char **argv) {
  int numero_cedula;
  ...
  numero_cedula = atoi(argv[1]);
  ...
}
```

y luego utilice el algoritmo del Ejercicio 6 para extraer los dígitos y guardarlos en el arreglo que toma la función en la primera parte.

Ejercicio 8 - Calculador de propinas (*)

Realizar un programa de nombre `propina` que calcule propinas y divida la cuenta entre los comensales. El programa debe ser invocado desde la línea de comandos de la siguiente forma

```
$./propina comensales comida porcentaje
```

donde `comensales` es el número (entero) de comensales, `comida` el costo (sin decimales) de la comida, y `porcentaje` es un valor entero entre 0 y 100 indicando el porcentaje del costo de la comida que se aplicará como propina.

Como resultado, el programa debe mostrar a pantalla algo similar a lo siguiente

```
Propina total: xxxx
Propina por persona: xxxx
Precio total a pagar: xxxx
Precio a pagar por persona: xxxx
```

NOTA: *Se sugiere utilizar la función `atoi` para convertir una cadena de texto a entero. Recuerde que para poder acceder a los argumentos en la línea de comandos, la función main debe ser declarada como `int main(char argc, char** argv)`; luego el argumento i-ésimo puede accederse como cadena de texto como `argv[i]`.*