



**ORGANIZACIÓN DE COMPUTADORAS**  
Departamento de Ciencias e Ingeniería de la Computación  
Universidad Nacional del Sur  
Segundo Cuatrimestre de 2017



<b>Primer Examen Parcial</b>		
Lic. en Ciencias de la Computación – Ing. en Computación – Ing. en Sistemas de Información		
Apellido y Nombre: (en ese orden)	LU:	Hojas entregadas: (sin enunciado)
Profesor:		
NOTA: Resolver los ejercicios en hojas separadas. Poner nombre, LU y número en cada hoja.		

*Apague cualquier dispositivo electrónico en su poder y manténgalo guardado. No puede utilizar auriculares, ni calculadora. Lea todo el ejercicio antes de comenzar a desarrollarlo.*

**Ejercicio 1.** Dado el número **decimal**  $-298,5625$  llevar adelante los siguientes cambios de base:

- a) Convertirlo a **octal**, empleando el método de la **división** tanto para la parte entera como para la parte fraccionaria, expresando el resultado en **complemento a la base**, con 4 dígitos octales para la parte entera y 6 para la parte fraccionaria.
- b) Convertirlo a **binario** utilizando el método de la **multiplicación** tanto para la parte entera como para la parte fraccionaria, expresando el resultado en **complemento a la base disminuida**, con 12 bits para la parte entera y 6 bits para la parte fraccionaria.

**Ejercicio 2.** Considerando los números **decimales**  $X = 1537$  e  $Y = 2559$ , llevar adelante las siguientes operaciones, indicando claramente el resultado obtenido y la existencia o no de *overflow*:

- a) Calcular  $-X - Y$ , trabajando en **hexadecimal** en **complemento a la base**, con una precisión de cuatro dígitos (incluido el signo).
- b) Calcular  $X + Y$ , trabajando en **hexadecimal** en **complemento a la base disminuida**, con una precisión de cuatro dígitos (incluido el signo).
- c) Calcular  $X - Y$ , haciendo uso de un hardware que opera en una codificación **BCD Exceso-3** y **complemento a la base**, con una precisión de cinco dígitos (incluido el signo), indicando claramente qué operación se está realizando en cada uno de los pasos intermedios.

**Ejercicio 3.** Considerando el Código Cíclico Redundante (CRC):

- a) Construir el mensaje  $T(x)$  a transmitir asociado al mensaje de datos  $M(x) = 110\,1011\,1011$  empleando el polinomio generador  $G(x) = x^4 + x + 1$ .
- b) Suponiendo que durante la transmisión el mensaje  $T(x)$  es modificado con un error  $E(x)$  de tal forma que el receptor recibe el mensaje  $T'(x) = 110\,0011\,0011\,1011$ , determinar cómo opera el mecanismo de detección de errores y cuál es la conclusión que se alcanza.

- c) Comparando el mensaje transmitido  $T(x)$  y el mensaje recibido  $T'(x)$ , ¿cuál es el desarrollo del polinomio de error  $E(x)$ ? Sabiendo cuál fue el error exacto que existió, ¿cuál es la longitud de la ráfaga en error? y ¿a qué conclusión se puede arribar?

**Ejercicio 4.** Considerando el código Hamming mínima distancia 4 (Hamming extendido), empleando paridad par y estando la secuencia ordenada de izquierda a derecha:

- Calcular los bits de código asociados al dato 0110 1011 y armar el codeword correspondiente que integra el dato y los bits calculados. ¿Cuántos bits de código se tienen que completar? Justifique su respuesta.
- Considerando que el receptor recibe el codeword 10111 1011 0110 que contiene los bits de dato y de código  $C_i$ . Recalcular los bits de código y determinar cuál es el síndrome.
- Determinar cómo trabaja el mecanismo de detección/corrección ante una política  $d = 2$ ,  $c = 1$ , con los resultados obtenidos en el inciso b).
- Determinar cómo trabaja el mecanismo de detección/corrección ante una política  $d = 3$ ,  $c = 0$ , si el síndrome fuera 1110.
- Determinar cómo trabaja el mecanismo de detección/corrección ante una política  $d = 2$ ,  $c = 1$ , si el síndrome fuera 0000.

**Ejercicio 5.** Dada la definición para TCadena, implementar en **lenguaje C**:

- Una función `int es_palindroma(TCadena cad)` que dada una cadena de caracteres `cad`, retorne 1 si `cad` es *palíndroma*, y cero en caso contrario. Una cadena de caracteres se dice *palíndroma*, si se lee de igual forma de izquierda a derecha que de derecha a izquierda. Ejemplo: “anana” y “neuquen”, son cadenas *palíndromas*.
- Una función `TCadena[] clonar (TCadena[] arr, int long)`, que dado un arreglo de cadenas de caracteres `arr` de longitud `long`, retorne un nuevo arreglo de cadenas de igual longitud que `arr`, pero que contenga la *clonación* de aquellas cadenas en `arr` que son *palíndromas*. Para esto, contemplar el uso de la función definida en el inciso anterior, así como un correcto uso de la función `malloc` para reservar memoria a la hora de clonar las cadenas.

Dado un *Arreglo A*, la función `clonar()` retornará un nuevo *Arreglo B*, tal como se indica en la siguiente figura:

```
typedef struct cadena{
    char * string;
    int longitud;
}* TCadena;
```

