UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Informática I

Exámen Final

17-Dic-2025

Apellido y Nombre	Legajo	# Hojas	Profesor

Lea atentamente el enunciado. Su correcta interpretación forman parte de la evaluación. Duración 90 minutos.

Parte Teórica

Dado el siguiente código:

```
#define FILS 4
#define COLS 4
int m[FILS][COLS] = {      {5, 6, 12, 56},
                           {78, -9, 67, -88},
                           {34, -2, 3, 45},
                           {-3, -99, 233, 66} };
int *p = m;
```

Suponiendo que la dirección donde se ubica **m** es **`0xbfff0000`** complete la siguiente tabla

1)	`*p`	
2)	`m[3][1]`	
3)	`*p+6`	
4)	`*(p+5)`	
5)	`&m[1][0]` (suponiendo enteros de 32 bits)	

6) ¿Cómo accedería al elemento de la tercera fila, primera columna usando m?

7) ¿Y usando p? Justificar generalizando para una matriz de **F**(filas) **x C**(columnas) si se quiere acceder al elemento de la fila **f** y la columna **c**.

8) Indique cuál es la salida en pantalla del siguiente código

```
#include <stdio.h>
#include <string.h>

int main(void) {
    char *p;
    strcpy(p, “Hola, mundo”);
    printf(“%s\n”, p);
    return 0;
}
```

9) Suponiendo que debe armar una lista simplemente enlazada para luego ordenarla y transmitir sus datos por un socket. ¿Qué tipo de estructuras utilizaría, a) ó b)? Justifique.

a)

```
struct dato_sensor{
       char   id[20];             // identificacion del sensor
       double temp;               // temperatura medida
};

struct nodo_sensor{
       struct dato_sensor *dato;  // puntero a los datos de un sensor
       struct nodo_sensor *prox;  // puntero al proximo sensor
};
typedef struct nodo_sensor sensor_t;
```

b)

```
struct nodo_sensor{
       char   id[20];             // identificacion del sensor
       double temp;               // temperatura medida
       struct nodo_sensor *prox;  // puntero al proximo sensor
};
typedef struct nodo_sensor sensor_t;
```

10) Complete las siguientes ecuaciones para obtener el resultado correspondiente

```
a) 0x67 & 0x55 =                          c) 0x64 ^ 0x36 =
b) 0x88 |      = 154                      d) ~0x77 | 3 =
```

Parte Práctica

Se pide desarrollar programa que procese datos obtenidos de un analizador lógico. Los datos están contenidos en un archivo que es el resultado de guardar en disco el contenido de una unión con el siguiente formato:

```
union logic_in {
    unsigned short int data;
    struct {
            unsigned int ch0  :1;
            unsigned int ch1  :1;
            unsigned int ch2  :1;
            unsigned int ch3  :1;
            unsigned int ch4  :1;
            unsigned int ch5  :1;
            unsigned int ch6  :1;
            unsigned int ch7  :1;
            unsigned int ch8  :1;
            unsigned int ch9  :1;
            unsigned int ch10 :1;
            unsigned int ch11 :1;
            unsigned int ch12 :1;
            unsigned int ch13 :1;
            unsigned int ch14 :1;
            unsigned int ch15 :1;
           } channel;
    };
```

Para realizar el procesamiento se solicita realizar las siguientes funciones:

1) void process_data(union logic_in *din, union logic_out *dout);
Recibe un puntero **din** a una unión del tipo **logic_in** y setea los bits correspondientes en la unión del tipo **logic_out** apuntada por **dout**. La unión **dout** tiene el siguiente formato:

```
union logic_out {
    unsigned char data;
    struct {
            unsigned int out0 :1;
            unsigned int out1 :1;
            unsigned int out2 :1;
            unsigned int out3 :1;
            unsigned int out4 :1;
            unsigned int out5 :1;
            unsigned int out6 :1;
            unsigned int out7 :1;
           } output;
    };
```

La lógica será la siguente:

- **`out0`** será 1 si todos los bits ch0, ch1, ch2 y ch3 son iguales entre sí. Es decir son todos 1 ó todos 0.
- **`out1`** será 1 si todos los bits ch4, ch5, ch6 y ch7 son iguales entre sí. Es decir son todos 1 ó todos 0.
- **`out2`** será 1 si todos los bits ch8, ch9, ch10 y ch11 son iguales entre sí. Es decir son todos 1 ó todos 0.
- **`out3`** será 1 si todos los bits ch12, ch13, ch14 y ch15 son iguales entre sí. Es decir son todos 1 ó todos 0.
- **`out4`** será 1 si todos los bits ch0, ch1, ch2 y ch3 son 0.
- **`out5`** será 1 si todos los bits ch4, ch5, ch6 y ch7 son 0.
- **`out6`** será 1 si todos los bits ch8, ch9, ch10 y ch11 son 0.
- **`out7`** será 1 si todos los bits ch12, ch13, ch14 y ch15 son 0.

`2) void compute_zeroes(union logic_in *d, int counts[]);`

Recibe un puntero a una unión del tipo **`logic_in`** y la dirección de comienzo de un array de enteros **`counts`**. Se asume que el array tiene un tamaño igual a la cantidad de canales que posee la estructura de campos de bits **`channel`** de la unión **`logic_in`** y que viene inicializado con cada uno de sus elementos en 0 la primera vez que se la invoca.

La función deberá incrementar el entero correspondiente a un canal si el bit de ese canal está en 0.

3) void print_stats(int counts[], int total);

Recibe la dirección de comienzo de un array de enteros **counts** y la cantidad total de registros procesados. Deberá imprimir en pantalla las estadísticas de cada uno de los canales con el porcentaje de unos que se obtuvo durante el procesamiento de los datos. El formato de salida será como el siguente:

```
Channel #0     50.13%
Channel #1     49.66%
Channel #2     49.92%
Channel #3     49.82%
Channel #4     49.78%
Channel #5     49.95%
Channel #6     49.89%
Channel #7     49.77%
Channel #8     49.79%
Channel #9     49.98%
Channel #10    50.35%
Channel #11    50.30%
Channel #12    49.73%
Channel #13    50.23%
Channel #14    50.02%
Channel #15    50.34%
```

4) ./plogic <arch_entrada> <arch_salida>

Realizar un programa que reciba por línea de comandos el nombre de una archivo de entrada con los datos provenientes del analizador lógico y el nombre de un archivo de salida donde salvar los datos procesados.

El programa deberá hacer los siguiente:
- Validar la cantidad de argumentos de la línea de comandos. En caso de error terminará con error **EARGS**.
- Abrir el archivo de entrada de donde leer los datos. En caso de error terminará con error **EINFILE**.
- Abrir el archivo de salidade donde se escribiran los datos procesados. En caso de error terminará con error **EOUTFILE**.
- Por cada uno de los datos leídos:
    1) se llamará a **process_data**
    2) se llamará a **compute_zeroes**
    3) se escribirá en el archivo de salida el dato procesado
    4) se incrementará el totalizador de muestras procesadas
- Una vez finalizado el procesamiento de todos los datos se llamará a **print_stats** para imprimir las estadísticas.
- El programa finaliza retornando **OK**

Notas

Podes utilizar todas las funciones de la librería estándar que consideres necesarias. También podes crear todas las funciones auxiliares que necesites. Se proverán los archivos de proyecto necesarios. Además se incluyen los archivos de datos para probar. **datasec.dat** posee una distribución secuencial por lo que el resultado es del 50% para cada uno de los bits y **datarand.dat** posee datos aleatorios.

El **Makefile** posee reglas para compilar el programa pricipal **plogic**.

```
practica/
        plogic.c
        func_plogic.c
        func_plogic.h
        datasec.dat
        datarand.dat
        Makefile
```