Parte Teórica

a) 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]`	

6) ¿cómo accedería al elemento de la cuarta fila, segunda columna usando m?

7) ¿y usando p? Justificar.

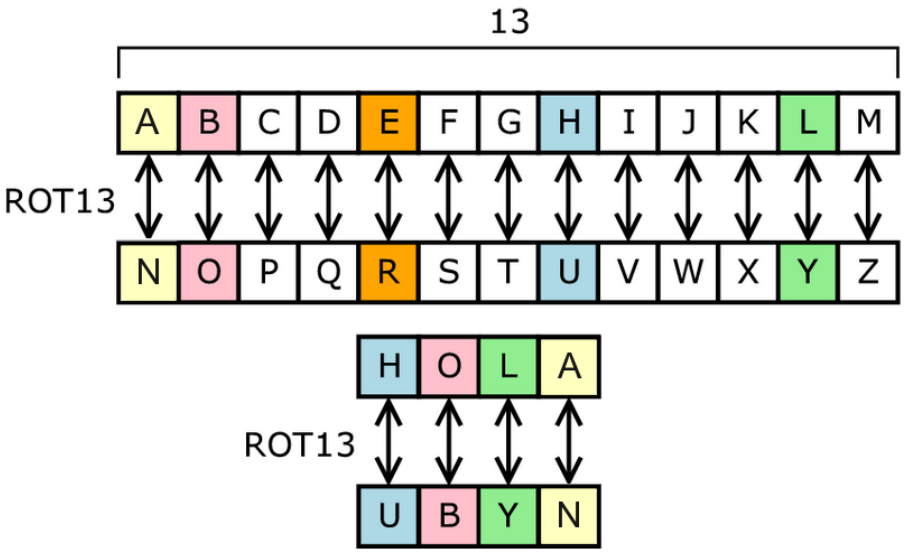
b) Explique cuál es la diferencia entre un proceso que está corriendo en primer plano (foreground) y uno que está corriendo en segundo plano (background).

c) ¿Qué es una librería estática? Ventajas y desventajas.

Parte Práctica

1) El algorítmo **ROT13**(«**rotar 13 posiciones**») es un sencillo algorítmo de cifrado utilizado para ocultar un texto sustituyendo cada letra por la letra que está trece posiciones por delante en el alfabeto. El agorítmo sólo afecta a las letras, tanto mayúsculas como minúsculas, pero no a los signos de puntuación, ni espacios en blanco, ni a los caracteres de control no imprimibles como \n, \t, etc.

El esquema de sustitución es el siguiente:



En forma idéntica se aplica para las letras minúsculas.

Se pide implementar la función **`int rot13(char *archivo);`** que recibe el nombre de un archivo de texto, por ejemplo “archivo.txt”, y genera un archivo “archivo_rot13.txt” cifrado con el arlgorítmo rot13.

Esta función retorna los siguientes valores:

 0 => salió todo bien.
-1 => error al procesar el archivo de entrada.
-2 => error al procesar el archivo de salida.

2) Dado el siguiente tipo de dato:

```
typedef struct {
                char palabra[20];
                int count;
               } sword_t;
```

Se pide implementar la función **`int contar_palabras(sword_t **a, char *archivo);`**
La función recibe la dirección de comienzo de un array de punteros a estructuras del tipo **`sword_t`** de longitud suficiente para contener todas las posibles palabras diferentes que puedan encontrarse al procesar el archivo y el nombre del archivo de texto a procesar. Originalmente el array viene inicializado con todos los punteros en **`NULL.`**

Esta función deberá procesar el achivo de texto separándolo en palabras, recorrer el array de punteros a estructuras buscando en cada estructura la aparición de dicha palabra. Si existe una estructura con dicha palabra se incrementará el contador “count”. Si no, se creará dinámicamente una nueva estructura para esa palabra, se copiará la palabra en el campo “palabra”, se inicializará el contador “count” en 1 y se agregará al array de punteros la referencia a la nueva estructura.

Esta función retorna los siguientes valores:

n => con n positivo indicando la cantidad de palabras distintas que contiene el archivo.
-1 => error al procesar el archivo de entrada.
-2 => error al reservar memoria para una nueva estructura.

3) Realizar un programa que reciba por linea de comandos nombres de achivos de texto. Por cada uno de estos archivos dispare un proceso hijo que ejecute el cifrado con el algorítmo rot13 implementado en el ejercicio 1. El programa deberá esperar la finalización de todos los procesos hijos para finalizar. Cada uno de los procesos hijos deberá indicar qué archivo está procesando y el estado después de la finalización del cifrado de acuerdo al resultado retornado por la función.