| Nº Orden | Apellido y nombre | L.U. | Cantidad de hojas |
|---|---|---|---|
| | | | |

# Organización del Computador 2
# Primer parcial – 20/05/2014

| 1 (40) | 2 (40) | 3 (20) | |
|---|---|---|---|
| | | | |

*Normas generales*

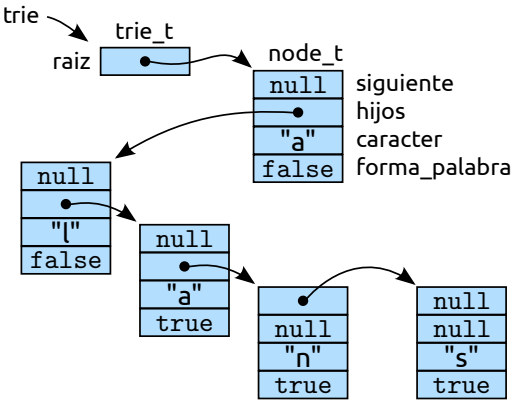
- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma **no** se incluye en la cantidad total de hojas entregadas.
- Está permitido tener los manuales y los apuntes con las listas de instrucciones en el examen. Está prohibido compartir manuales o apuntes entre alumnos durante el examen.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido y LU.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen tres notas: I (Insuficiente): 0 a 59 pts, A- (Aprobado condicional): 60 a 64 pts y A (Aprobado): 65 a 100 pts. No se puede aprobar con A- ambos parciales. Los recuperatorios tienen dos notas: I: 0 a 64 pts y A: 65 a 100 pts.

## Ej. 1. (40 puntos)

Considerando una estructura de trie como la siguiente:

```
typedef struct trie_t {
        nodo *raiz;
} __attribute__((packed)) trie;

typedef struct nodo_t {
        struct nodo_t *sig;
        struct nodo_t *hijos;
        char c;
        bool fin;
} __attribute__((packed)) nodo;
```




donde `bool` equivale a 1 `byte`.

La estructura almacena letras, las mismas están ordenadas lexicográficamente por *nivel* del trie. Un *nivel* está compuesto por los nodos que se encuentran enlazados a través del campo `sig`. Lamentablemente, la implementación que está en producción tiene un error muy difícil de encontrar, en el cual resulta ser que las letras no quedan ordenadas correctamente por nivel.

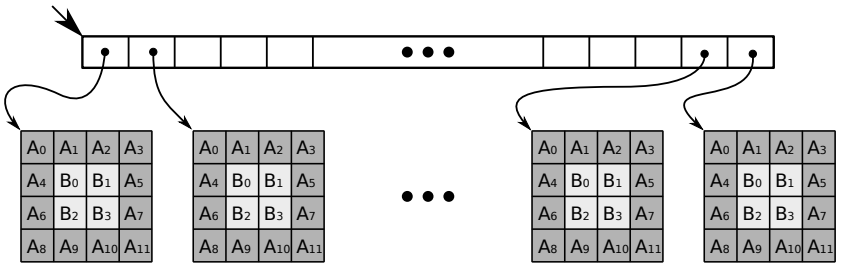
Se requiere desarrollar una función que dado un trie compruebe que las letras en todos los niveles están ordenadas lexicográficamente. En caso de que encuentre un nivel no ordenado, debe retornar el **doble puntero** al primer nodo del nivel desordenado, en caso contrario devolverá `null`. La aridad de la función será: `nodo** check_trie(trie* t)`

(a) **(10p)** Implementar en `lenguaje C` la función `check_trie`.

(b) **(30p)** Implementar en `ASM` la función `check_trie` utilizando como pseudo-código la implementación en `lenguaje C` del punto anterior.

# Ej. 2. (40 puntos)

Sea la siguiente estructura que almacena un vector de punteros a matrices como indica la figura. Cada matriz tiene $4 \times 4$ elementos de 1 byte cada uno, tomados como numeros sin signo.




Cada matriz en la figura está dividida en dos secciones, A y B. Construir una función que calcule el promedio de todos los elementos indicados por la letra A.

La aridad de la función es: `double promediosA(int n, char** matrices)`
Donde `n` es la cantidad de elementos del vector y `matrices` es el puntero al vector de matrices.

(a) **(20p)** Implementar el código ASM necesario para el cálculo del promedio para una sola matriz en paralelo suponiendo que `rax` contiene el puntero a la matriz.

(b) **(20p)** Utilizando el ejercicio anterior **como guia**, escribir el código de `double promediosA(int n, char** matrices)`

Notas: La suma total entra en un entero sin signo de 32bits.
Se debe recorrer una sola vez el vector de punteros.

# Ej. 3. (20 puntos)

Dada una lista doblemente encadenada de `n` elementos con la siguiente estructura:

```
typedef struct list_nodo_t {
        struct list_nodo_t *siguiente;
        struct list_nodo_t *anterior;
        char c;
} __attribute__((packed)) list_nodo;
```

(a) **(20p)** Escribir una función en ASM que recorra la lista generando un vector con los elementos de la misma y llame a la función `void send_list(char* vector)` con el vector resultante. La aridad de la función será `void send_list(list_nodo* ln)`.

**Importante:** Esta función no debe usar memoria dinámica, sólo está permitido almacenar el vector en la pila. Además, el orden de los elementos en el vector debe ser el mismo que en la lista.