Apellido, Nombre:

.....

Mail:

.....

Padrón:

.....

Teórico / Práctico - Entrego ..... hojas			
1:	2:	3:	4: .....

Nota Final
.....

Aclaraciones:

- **Antes de comenzar a resolver el parcial, complete sus datos en esta hoja, y al finalizarlo, firme todas las hojas.**
- **Se deben numerar TODAS las hojas e inicializarlas con nombre, apellido, padrón.**
- **La aprobación del parcial está sujeta a la correcta realización de al menos el 50% de 3 ejercicios y sumar 10 puntos en total como mínimo.**

1) Aritmética de punteros / Memoria Dinámica (5 puntos)

En este juego de tronos tan cambiante, todos buscan obtener ventajas aún cuando sea traicionando a los demás y Daenerys Targaryen no perdona a quienes la traicionan, que han sido y serán muchos.
Le llegó a sus manos un algoritmo que, según le afirmaron, tiene el nombre de alguien que la traicionó....

```
const unsigned int MAX_ABC = 26;
const int LETRA_A = 65;

int main(){
	int pos_letras[9] = {3, 17, 14, 11, 18, 24, 17, 0, 21};

	char* abecedario = malloc(sizeof(char)*MAX_ABC);
	for(int i = 0; i < MAX_ABC; i++)
		abecedario[i] = (char)(LETRA_A + i);

	char*** frase = malloc(sizeof(char*)*9);

	for(int i = 0; i < 9; i++){
		if (i < 4){
			frase[i] = malloc(sizeof(char*)*(size_t)(5-i));
			for(int j = 0; j < (5-i); j++)
				frase[i][j] = &(abecedario[pos_letras[i]]);
		} else {
			frase[i] = malloc(sizeof(char*)*(size_t)(i-3));
			for(int j = 0; j < (i-3); j++)
				frase[i][j] = &(abecedario[pos_letras[i]]);
		}
	}

	for(int i = 3; i >= 0; i--) printf("%c", frase[i][0][0]);
	for(int i = 8; i >= 4; i--) printf("%c", frase[i][0][0]);

	return 0;
}
```

a. Realizar los diagramas del stack y del heap. Mostrar que se imprime por pantalla.
b. El algoritmo no libera la memoria, crear la parte que falta.

"Hay que tener imaginación para cambiar la imaginación." - Luis Pescetti.

2) Análisis de Algoritmos (5 puntos)

1. El **Teorema Maestro** es el abc para hallar el tiempo de ejecución de algoritmos que utilizan divide y conquista, como herramienta de diseño de algoritmos:
    a. Enunciar el teorema.
    b. Proveer una ecuación ejemplo para cada caso que presenta el teorema.

2. Dadas las implementaciones con memoria dinámica de los tda pilas, colas y lista hacer un gráfico comparativo del tiempo de ejecución en notación Big-O de cada operación del tda, explique en qué implementación se basa para cada TDA.
   ¿Qué tda es el mejor?

Operación	pila	cola	lista
crear			
insertar			
insertar_pos			
borrar			
borrar_pos			
destruir			

3. Calcular el tiempo de ejecución de :
    a. $T(n) = 3T(n/3) + n/2$
    b. $T(n) = 7T(n/2) + n^2$
    c. $T(n) = 64T(n/8) - n^2$

3) TDA (5 puntos)

Westeros es, en superficie, equivalente a Sudamérica, a lo largo de su extensión podemos encontrar muchas casas. Cada casa, tiene un nombre, un lema y un ejército. En estos tiempos de guerra, donde todo es dominación y traiciones, la geografía de Westeros cambia muy rápido.
Dados los siguientes structs:

```
typedef struct casa {
        char* nombre;
        char* lema;
        int ejercito;
} casa_t;

typedef struct westeros {
        casa_t* casas;
        int cantidad_casas;
} westeros_t;
```

"Hay que tener imaginación para cambiar la imaginación." - Luis Pescetti.

Se pide implementar las siguientes funciones del tda **westeros**:

```
/*
 * Dará de alta una casa en westeros, ampliando la memoria reservada.
 * Devuelve 0 si pudo o -1 si no pudo o -2 si ya existe la casa.
 */
int westeros_nueva_casa(westeros_t* westeros, char* nombre, char* lema, int ejercito);

/*
 * La casa dominante intentará dominar a la casa dominada.
 * Si ambas existen y el ejército de la dominante es mayor al de la dominada
 * entonces la dominada deberá desaparecer, y la dominante aumentará
 * su ejército a la mitad (porque muchos soldados mueren vio...).
 * Si ambas existen y el ejército de la dominante es menor o igual al de la dominada
 * entonces el ejército de la dominante bajará a la mitad (algunos huyen...).
 * Se deberá liberar la memoria en caso de dominar la casa.
 * Devolverá 0 si pudo dominarse la casa o -1 si no, si alguna de las casas no
 * existe, devolverá -2.
 */
int westeros_dominar_casa(westeros_t* westeros, char* dominante, char* dominada);
```

4) Recursividad (5 puntos)

Dado el siguiente algoritmo:

```
#include <stdio.h>
#include "pila.h"

void algo(char* cosa, pila_t* pila){
	if(*cosa == '\0')
		return;
	pila_apilar(pila, cosa);
	algo(cosa+1, pila);
	pila_apilar(pila, cosa+1);
}

int main(){
	pila_t* pila = pila_crear();
	char* cosa = "?AB2LMOG";

	algo(cosa, pila);

	for(int i=0;!pila_vacia(pila);i++){
		if((i%3) == 0) //i es 0..3..6..9..etc
			printf("%c", *(char*)pila_tope(pila));
		pila_desapilar(pila);
	}
	pila_destruir(pila);
	return 0;
}
```

a. Dibujar el contenido de la pila luego de ejecutar **algo**.
b. Indicar qué se imprime por pantalla.