~~~ SI CREES QUE ESTO TIENE UN FINAL FELIZ, NO HAS ESTADO PRESTANDO ATENCION ~~~

APELLIDO, NOMBRE: ..... PADRON: .....

MAIL: ..... ENTREGO ..... HOJAS.

1.a	1.b	2	3.a	3.b	4	NOTA

Antes de empezar a resolver el examen lea las siguientes aclaraciones:

- Complete sus datos en esta hoja. Firme, numere e inicialice con nombre, apellido y padrón todas sus hojas.
- Léalo **todo** a conciencia, y haga preguntas sobre lo que no entiende en el espacio designado para ello.
- Recomendamos fuertemente realizar un análisis de **cada** ejercicio.
- Para aprobar es necesario tener bien, al menos, el 60% de todo el examen.

EJERCICIOS

1. **Análisis de algoritmos**

   a. Sean:

   - T1(n) = 3T(n/2) + n2
   - T2(n) = 4T(n/2) + n2

   Determinar y demostrar cuál de los dos algoritmos tiene mejor tiempo de ejecución.

   b. Determinar el orden de crecimiento de la siguiente función:

```
void una_funcion(int n){
      int i, j, k, g, contador=0;
      for(i=0; i<=n; i++){
            for(j=0; j<=n; j*=i)
                  contador++;
            if (n%2==0)
                  for(g=n; g>=0; g--)
                        contador++;
            else
                  for(k=1; k<n; k*=2)
                        contador++;
      }
}
```

2. **Ordenamientos**

   a. Dado el siguiente vector:

a	c	d	g	e	r	y	t	f	h	z	c	b	n	m	w	u	i

   Ordenarlo ascendentemente mediante **quicksort** y **mergesort** mostrando todos los pasos intermedios.
~~~

3. **Aritmética de punteros**
   a. ¿Cuál o cuáles de los siguientes algoritmos es/son correctos? Grafique cada uno.

```
/***** 1 *****/
int main(){
	int a = 4;
	int* b;
	b = &a;
}

/***** 2 *****/
int main(){
	int* b;
	int a = 4;
	b = &a;
}

/***** 3 *****/
int main(){
	int a = 4;
	int* b;
	*b = a;
}
```

```
/***** 4 *****/
int main(){
	int a = 4;
	int* b;
	b = &a;
}

/***** 5 *****/
int main(){
	int* b;
	int a = 4;
	*b = a;
}

/***** 6 *****/
int main(){
	int a = 4;
	int* b;
	&b = a;
}
```

   b. Dado el siguiente algoritmo, diagrame como se crean y llenan las estructuras. Libere la memoria reservada.

```
int main(){
	int*** matriz = malloc(3*sizeof(int*));
	for(int i = 0; i < 3; i++){
		matriz[i] = malloc(sizeof(int*));
	}

	for(int i = 0; i < 9; i++){
		int* numero = malloc(sizeof(int));
		*numero = 9-i;
		matriz[i/3][i%3] = numero;
	}

}
```

4. **Recursividad**

Cree una rutina **recursiva** que reciba una cola y 2 pilas. Desencole los elementos y llene ambas pilas, una con los elementos al derecho y otra con los elementos al reves de como estan en la cola.

Cola

1	6	8	4	1	2

Pila 1

1	6	8	4	1	2

Pila 2

2	1	4	8	6	1