# Algoritmos y Estructuras de datos
# Ejercicios previos al primer examen parcial

## Problemas de recursión

- Escribe una implementación recursiva de la función `imprime_mitades`, la cual toma un entero no negativo *n* e imprime la secuencia *n*, *n/2*, *n/4*, ..., *1*, *0* (sin las comas). Por ejemplo, para *n*=9 la función debe imprimir 9 4 2 1 0 mientras que para *n*=0 sólo debe imprimir 0. Está prohibido usar ciclos y la biblioteca de C++ excepto las rutinas de escritura (`printf`, `cout`).

```
void imprime_mitades(int n) {
   std::cout << n << " ";
   if (n != 0) {
      imprime_mitades(n / 2);
   }
}
```

- Escribe una implementación recursiva de la función `imprime_alternado`, la cual toma un entero no negativo *n* e imprime los primeros *n* números de la secuencia 1, 0, 1, 0, 1, ... (sin las comas). Por ejemplo, para *n*=3 la función debe imprimir 1 0 1 mientras que para *n*=0 no debe imprimir nada. Está prohibido usar ciclos y la biblioteca de C++ excepto las rutinas de escritura (`printf`, `cout`).

```
void imprime_alternado(int n) {
   if (n != 0) {
      imprime_alternado(n - 1);
      std::cout << n % 2 << " ";
   }
}
```

- Escribe una implementación recursiva de la función `imprime_escalera`, la cual toma dos enteros no negativos *k*, *n* (donde $k \leq n$) e imprime la secuencia *k*, *k+1*, ..., *n-1*, *n*, *n-1*, ..., *k+1*, *k* (sin las comas). Por ejemplo, para *k*=2, *n*=5 la función debe imprimir 2 3 4 5 4 3 2. Está prohibido usar ciclos y la biblioteca de C++ excepto las rutinas de escritura (`printf`, `cout`).

```
void imprime_escalera(int k, int n) {
   std::cout << k << " ";
   if (k < n) {
      imprime_escalera(k + 1, n);
      std::cout << k << " ";
   }
}
```

- Escribe una implementación recursiva de la función `logaritmo2`, la cual toma un entero *n* potencia de 2 y devuelve el logaritmo base 2 de *n*. Por ejemplo, para *n*=32 la función debe devolver 5. Está prohibido usar ciclos y la biblioteca de C++.

```
int logaritmo2(int n) {
   if (n == 1) {
      return 0;
   } else {
      return 1 + logaritmo2(n / 2);
   }
}
```

# Problemas sobre secuencias

- Escribe una implementación de la función `es_palindromo`, la cual toma una secuencia de caracteres y devuelve verdadero si la cadena correspondiente es palíndroma y falso en otro caso. No puedes usar ninguna utilidad de la biblioteca de C++.

```
bool es_palindromo(char* ini, char* fin) {
   while (fin - ini >= 2) {
      if (*ini != *(fin - 1)) {
         return false;
      }
      ++ini, --fin;
   }
   return true;
}
```

- Escribe una implementación de la función `es_ordenada`, la cual toma una secuencia de enteros y devuelve verdadero si la secuencia está ordenada de menor a mayor y falso en otro caso. Por ejemplo, la secuencia { 1, 2, 2, 3 } sí está ordenada mientras que la secuencia { 1, 2, 3, 2 } no lo está. No puedes usar ninguna utilidad de la biblioteca de C++.

```
bool es_ordenada(int* ini, int* fin) {
   for (int* p = ini; p < fin && p + 1 < fin; ++p) {
      if (*p > *(p + 1)) {
         return false;
      }
   }
   return true;
}
```

- Escribe una implementación de la función `intercambia_parejas`, la cual toma una secuencia de enteros e intercambia los elementos de sus parejas disjuntas contiguas, comenzando por el inicio. Si la secuencia tiene un número impar de elementos, el último elemento no debe modificarse. Por ejemplo, el resultado de aplicar la función a la secuencia { 1, 2, 3, 4, 5, 6 } es { 2, 1, 4, 3, 6, 5 } y el resultado para la secuencia { 1, 2, 3, 4, 5 } es { 2, 1, 4, 3, 5 }. Puedes usar las utilidades de `<algorithm>` de la biblioteca de C++.

```
void intercambia_parejas(int* ini, int* fin) {
   for (int* p = ini; p < fin && p + 1 < fin; p += 2) {
      std::swap(*p, *(p + 1));
   }
}
```

- Escribe una implementación de la función `intercambia_mitades`, la cual toma una secuencia de enteros y la modifica intercambiando sus dos mitades. Si la secuencia tiene un número impar de elementos, el elemento de enmedio debe permanecer en su lugar. Por ejemplo, el resultado de aplicar la función a la secuencia { 1, 2, 3, 4 } es { 3, 4, 1, 2 } y el resultado para la secuencia { 1, 2, 0, 3, 4 } es { 3, 4, 0, 1, 2 }. Puedes usar las utilidades de `algorithm` de la biblioteca de C++.

```
void intercambia_mitades(int* ini, int* fin) {
   int* mitad = ini + (fin - ini) / 2 + (fin - ini) % 2;
   while (mitad < fin) {
      std::swap(*ini, *mitad);
      ++ini, ++mitad;
   }
}
```