# Algoritmos y Estructuras de datos
# Posibles soluciones a los ejercicios previos al primer examen parcial

En el examen no importa si sus soluciones son diferentes, siempre y cuando funcionen.

- Escribe una implementación recursiva de la función `imprime_escalera`, la cual toma dos enteros i, n e imprime la secuencia i, i+1, ..., n-1, n, n-1, ..., i+1, i (sin las comas). Por ejemplo, para i=2, n=5 la función debe imprimir 2 3 4 5 4 3 2. Puedes asumir que i ≤ n.

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

- Escribe una implementación recursiva 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) {
   if (fin - ini <= 1) {
      return true;
   } else {
      return *ini == *(fin - 1) && es_palindromo(ini + 1, fin – 1);
   }
}
```

- Escribe una implementación recursiva de la función `is_sorted`, 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 is_sorted(int* ini, int* fin) {
   if (fin - ini <= 1) {
      return true;
   } else {
      return *ini <= *(ini + 1) && is_sorted(ini + 1, fin);
   }
}
```

- 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 debe permanecer sin modificaciones. 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 asumir que están disponibles todas las funciones de la biblioteca de C++.

```
void intercambia_parejas(int* ini, int* fin) {
   for (int* p = ini; fin - p >= 2; 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 asumir que tienes disponibles únicamente las utilidades de `algorithm` de la biblioteca de C++.

```
void intercambia_mitades(int* ini, int* fin) {
   int t = fin - ini, m = t / 2;
   for (int i = 0; i < m; ++i) {
      std::swap(*(ini + i), *(ini + i + m + t % 2));
   }
}
```