3.function calculateDiscountedPrice(price, tier) {

const discounts = {

GOLD: 0.8,

SILVER: 0.9,

};

const discountFactor = discounts[tier] ?? 1;

return price \* discountFactor;

}

Alasan perubahan:

1. Nama fungsi dan parameter jelas
   * d diganti jadi calculateDiscountedPrice supaya jelas fungsinya.
   * p jadi price, t jadi tier supaya parameter mudah dimengerti.
2. Menggunakan objek untuk mapping diskon
   * Mempermudah penambahan atau perubahan diskon di masa depan hanya dengan mengedit objek discounts.
   * Menghindari if-else yang berulang dan membuat kode lebih ringkas.
3. Menggunakan operator nullish coalescing ??
   * Jika tier tidak ditemukan, diskon default 1 (tidak ada potongan).
4. Lebih mudah maintenance dan scalable
   * Kalau perlu tambah tier baru, tinggal tambah property baru di objek discounts.

4.function findPairsWithSum(array $nums, int $target): array {

$seen = [];

$usedPairs = [];

$result = [];

foreach ($nums as $num) {

$complement = $target - $num;

if (isset($seen[$complement])) {

$pair = [min($num, $complement), max($num, $complement)];

$key = implode('-', $pair);

if (!isset($usedPairs[$key])) {

$result[] = $pair;

$usedPairs[$key] = true;

}

}

$seen[$num] = true;

}

return $result;

}

$input = [1, 2, 3, 4, 5];

$target = 6;

$pairs = findPairsWithSum($input, $target);

print\_r($pairs);

Jadi, fungsi ini kerjanya:

* jalanin satu-satu angka di array.
* Buat tiap angka, kita cari angka lain yang kalau dijumlahin sama targetnya.
* Nah, supaya gak ketemu pasangan yang sama dua kali (kayak [1,5] sama [5,1]), kita simpan pasangan itu dalam bentuk yang sudah diurutkan kecil-besar.
* Kalau pasangan itu belum pernah kita temuin, baru kita masukin ke hasil.
* Gampangnya, kita pakai semacam “catatan” angka yang sudah kita lihat dan pasangan yang sudah pernah kita ambil, supaya gak dobel.

Jadi intinya: cek setiap angka, lihat apa ada temannya yang bikin jumlahnya pas target, dan jangan sampe hasilnya ada yang dobel.