```python
def fix_mismatch(mismatch, power, min_limit, max_limit):
    """
    func fix_mismatch :: Real, [Real], [Real], [Real] -> [Real]

    change the total generated power by `mismatch`. Do this based upon current
    power of each generator taking into account its limits.
    Returns a list of new generator powers
    """

    assert(len(power) == len(min_limit) == len(max_limit))

    if mismatch == 0: return power

    # make sure we have capacity for mismatch
    assert sum(min_limit) < sum(power) + mismatch < sum(max_limit)

    done = [False for _ in range(len(power))]
    result = [0.0 for _ in range(len(power))]

    def find_limit_max(m):
        """find the index of the first generator that will
        be limited. or None """
        for n in range(len(done)):
            if (not done[n]) and (power[n] * m > max_limit[n]):
                return n
        return None

    def find_limit_min(m):
        """find the index of the first generator that will
        be limited. or None """
        for n in range(len(done)):
            if (not done[n]) and (power[n] * m < min_limit[n]):
                return n
        return None

    # deal with each generator that will be limited
    while True:
        assert(not all(done))

        total_gen = sum(power[i] for i in range(len(done)) if not done[i])
        assert(total_gen != 0)

        multiplier = 1.0 + (mismatch / total_gen)

        if mismatch < 0:
            idx_gen = find_limit_min(multiplier)
            if idx_gen is None: break

            # generator hit min limit: idx_gen
            result[idx_gen] = min_limit[idx_gen]
            mismatch -= result[idx_gen] - power[idx_gen]
            done[idx_gen] = True
        else:
            idx_gen = find_limit_max(multiplier)
            if idx_gen is None: break

            # generator hit max limit: idx_gen
            result[idx_gen] = max_limit[idx_gen]
            mismatch -= result[idx_gen] - power[idx_gen]
            done[idx_gen] = True

    # deal with all the other generators knowing that none of them will limit
    for idx in range(len(power)):
        if not done[idx]:
            result[idx] = power[idx] * multiplier
            mismatch -= result[idx] - power[idx]
            done[idx] = True

    # check nothing is out of limits
    for idx in range(len(power)):
        assert(min_limit[idx] <= power[idx] <= max_limit[idx])
    assert mismatch < 0.001
    assert all(done)
    return result
```