```python
from doctest import testmod

pieces = [100, 50, 20, 10, 5, 2, 1]

def rendu_glouton(arendre, solution=[], i=0):
    """ Calcul du rendu de monnaie avec un algorithme récursif.

    Args:
        arendre (int): valeur dont on cherche à rendre la monnaie
        solution (list, optional): solution trouvée jusqu'à présent. Defaults to [].
        i (int, optional): rang de la pièce courante. Defaults to 0.

    Returns:
        list: tableau de somme d'argent, pièce par pièce

    Tests et Exemples:
    >>> rendu_glouton(68, [], 0)
    [50, 10, 5, 2, 1]
    >>> rendu_glouton(291, [], 0)
    [100, 100, 50, 20, 20, 1]
    """
    if arendre == 0:
        # cas de base de l'algo récursif
        return solution

    p = pieces[i]
    if p <= arendre :
        # il est possible d'utiliser la pièce courante p
        solution.append(p)
        return rendu_glouton(arendre - p, solution, i)
    else :
        # la pièce courante p est trop grande, il faut changer de pièce
        return rendu_glouton(arendre, solution, i + 1)


assert rendu_glouton(68, [], 0) == [50, 10, 5, 2, 1]
assert rendu_glouton(291,[],0) == [100, 100, 50, 20, 20, 1]

testmod()
```