**Minimum number of Coins**

**Medium**

Given an **infinite supply** of each denomination of Indian currency { **1, 2, 5, 10, 20, 50, 100, 200, 500, 2000** } and a target value **N**.  
Find the **minimum** number of coins and/or notes needed to make the change for Rs **N**. You must return the list containing the value of coins required.

**Example 1:**

**Input:** N = 43

**Output:** 20 20 2 1

**Explaination:**

Minimum number of coins and notes needed

to make 43.

**Example 2:**

**Input:** N = 1000

**Output:** 500 500

**Explaination:** minimum possible notes

is 2 notes of 500.

**Your Task:**  
You do not need to read input or print anything. Your task is to complete the function **minPartition()** which takes the value N as input parameter and returns a list of integers in decreasing order.

**Expected Time Complexity:** O(N)  
**Expected Auxiliary Space:**O(N)

**Constraints:**  
1 ≤ N ≤ 106

**Company Tags**

[**Visa**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Visa)

//{ Driver Code Starts

// Initial Template for Java

import java.io.\*;

import java.util.\*;

class CodingMaxima{

public static void main(String args[])throws IOException

{

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(in.readLine());

while(t-- > 0){

int N = Integer.parseInt(in.readLine());

Solution ob = new Solution();

List<Integer> numbers= new ArrayList<Integer>();

numbers = ob.minPartition(N);

for(int i: numbers){

System.out.print(i + " ");

}

System.out.println();

}

}

}

// } Driver Code Ends

// User function Template for Java

class Solution{

static List<Integer> minPartition(int N)

{

ArrayList<Integer> list=new ArrayList<Integer>();

int[] arr={2000, 500, 200,100,50, 20,10,5,2,1};

for(int i=0;i<arr.length;i++){

while(N>=arr[i] && N>0){

N = N - arr[i];

list.add(arr[i]);

}

if(N == 0) break;

}

//2nd method

// while(N!=0){

// if(N%2000==0){

// list.add(2000);

// N=N-2000;

// }

// else if(N%500==0){

// list.add(500);

// N=N-500;

// }

// else if(N%200==0){

// list.add(200);

// N=N-200;

// }

// else if(N%100==0){

// list.add(100);

// N=N-100;

// }

// else if(N%50==0){

// list.add(50);

// N=N-50;

// }

// else if(N%20==0){

// list.add(20);

// N=N-20;

// }

// else if(N%10==0){

// list.add(10);

// N-=10;

// }

// else if(N%5==0){

// list.add(5);

// N-=5;

// }

// else if(N%2==0){

// list.add(2);

// N=N-2;

// }

// else if(N%1==0){

// list.add(1);

// N=N-1;

// }

// }

// Collections.reverse(list);

return list;

}

}