Problem
In 2016, it was shown that every positive integer can be written as the sum of three or fewer palindromic terms. For t e purposes of this problem, a palindromic term is a string of digits (with no leading zeroes) that represents a positive nteger and reads the same forward and backward.

Given a positive integer S, find K palindromic terms that sum to S, such that K is minimized.

Input
The first line of input gives the number of test cases, T. T lines follow, each containing a positive integer S.

Output
For each test case, output one line of the form Case #x: A1 (if only one term is needed), Case #x: A1 A2 (if only two terms are needed), or Case #x: A1 A2 A3 (if three terms are needed), where x is the case number (counting starting f om 1), each Ai is a palindromic term (as described above), and the sum of the Ais equals S.

Limits
Time limit: 20 seconds per test set.
Memory limit: 1GB.
1 ≤ T ≤ 100.

Test set 1 (Visible)
1 ≤ S ≤ 1010.

Test set 2 (Hidden)
1 ≤ S ≤ 1040.

Sample

Input

Output

```
3
1
198
1234567890
```

```
Case #1: 1
Case #2: 191 7
Case #3: 672787276 94449 561686165
```

In Sample Case #1, the input is already a palindrome.

In Sample Case #2, note that 99 99, for example, would also be an acceptable answer. Even though there are multipl instances of 99, they count as separate terms, so this solution uses the same number of terms as 191 7.

Also note that 191 07, 181 8 9, 0110 88, 101 97, 7.0 191.0, and -202 4, for example, would not be acceptable answer .

Solution:

```
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include <vector>
#include <deque>
#include <queue>
#include <stack>
#include <set>
#include <map>
#include <algorithm>
#include <functional>
#include <utility>
#include <bitset>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <cstdio>

using namespace std;

#define REP(i,n) for((i)=0;(i)<(int)(n);(i)++)
#define snuke(c,itr) for(__typeof((c).begin()) itr=(c).begin();itr!=(c).end();itr++)

typedef long long ll;

vector <ll> palinds;

void dfs_pre(string s){
 int n = s.length();
 int i;

 if(n >= 1 && s[0] != '0'){
  ll x = 0;
  REP(i,n) x = x * 10 + (s[i] - '0');
  palinds.push_back(x);
 }

 if(n <= 8) for(char c='0';c<='9';c++) dfs_pre(c + s + c);
}

void pre(void){
 dfs_pre("");
 for(char c='0';c<='9';c++) dfs_pre((string)("") + c);
 sort(palinds.begin(),palinds.end());
// cout << palinds.size() << endl;
}

pair <ll, ll> find_two(ll X){
 int i,j;
 int N = palinds.size();

 j = N-1;

 REP(i,N){
  while(j >= 0 && palinds[i] + palinds[j] > X) j--;
  if(j >= 0 && palinds[i] + palinds[j] == X) return make_pair(palinds[i], palinds[j]);
 }

 return make_pair(-1ll, -1ll);
}

void main2(void){
 ll X;
 cin >> X;

 int i;
 int N = palinds.size();

 REP(i,N) if(palinds[i] == X){
  cout << X << endl;
  return;
 }

 pair <ll, ll> p = find_two(X);
 if(p.first != -1){
  cout << p.first << ' ' << p.second << endl;
  return;
 }

 int cnt = 0;
 REP(i,N) if(palinds[i] < X) cnt++;

 while(1){
  ll z = palinds[rand() % cnt];
  p = find_two(X - z);
  if(p.first != -1){
   cout << z << ' ' << p.first << ' ' << p.second << endl;
   return;
  }
 }
}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int main(void){
 pre();
 int TC,tc;
 cin >> TC;
 REP(tc,TC){
  printf("Case #%d: ", tc + 1);
  main2();
 }
 return 0;
}
```