// two methods

#include<bits/stdc++.h>

using namespace std;

void solve1(int n, int \*arr){

unordered\_map<int,bool> mp;

//making all values of mp as true

for(int i=0;i<n;i++){

int x = arr[i];

mp[x] = true;

}

//check for x-1 element . if x-1 doesn't exist means that x is the first element of the sequence.

//now make y = x+1 and check for all the subsequence element and len++. record the max len

//and starting point of the list.

int max\_len = INT\_MIN;

int max\_sub;

for(int i=0;i<n;i++){

int x = arr[i];

int len = 1;

if(mp.find(x-1)==mp.end()){

int y = x+1;

while(mp.find(y)!=mp.end()){

len++;

y = y+1;

}

if(y-x>max\_len){

max\_len = y-x;

max\_sub = x;

}

}

}

for(int i=0;i<max\_len;i++){

cout<<max\_sub+i<<endl;

}

}

void solve(int n, int \*arr){

unordered\_map<int,bool> mp;

//making all values of mp as true

for(int i=0;i<n;i++){

int x = arr[i];

mp[x] = true;

}

// making only those vale true which is lowest value in subsequence

for(int i=0;i<n;i++){

int x = arr[i];

if(mp.find(x-1)!=mp.end()){

mp[x] = false;

}

}

// just take the elements and note if they are the max subsequence in the given list

int max\_len = INT\_MIN;

int max\_sub;

for(int i=0;i<n;i++){

int x = arr[i];

if(mp[x]==true){

int len = 1;

int tsp = x;

while(mp.find(tsp+len)!=mp.end()){

len++;

}

if(len>max\_len){

max\_sub = tsp;

max\_len = len;

}

}

}

for(int i=0;i<max\_len;i++){

cout<<max\_sub+i<<endl;

}

}

int main(){

int n;

cin>>n;

int arr[n];

for(int i=0;i<n;i++){

cin>>arr[i];

}

solve(n,arr);

}