Given a number, find the next smallest palindrome

Given a number, find the next smallest palindrome larger than this number. For example, if the input number is “2 3 5 4 5″, the output should be “2 3 6 3 2″. And if the input number is “9 9 9″, the output should be “1 0 0 1″.

The input is assumed to be an array. Every entry in array represents a digit in input number. Let the array be ‘num[]’ and size of array be ‘n’

There can be three different types of inputs that need to be handled separately.
1) The input number is palindrome and has all 9s. For example “9 9 9″. Output should be “1 0 0 1″
2) The input number is not palindrome. For example “1 2 3 4″. Output should be “1 3 3 1″
3) The input number is palindrome and doesn’t have all 9s. For example “1 2 2 1″. Output should be “1 3 3 1″.

Solution for input type 1 is easy. The output contains n + 1 digits where the corner digits are 1, and all digits between corner digits are 0.

Now let us first talk about input type 2 and 3. How to convert a given number to a greater palindrome? To understand the solution, let us first define the following two terms:
Left Side: The left half of given number. Left side of “1 2 3 4 5 6″ is “1 2 3″ and left side of “1 2 3 4 5″ is “1 2″
Right Side: The right half of given number. Right side of “1 2 3 4 5 6″ is “4 5 6″ and right side of “1 2 3 4 5″ is “4 5″
To convert to palindrome, we can either take the mirror of its left side or take mirror of its right side. However, if we take the mirror of the right side, then the palindrome so formed is not guaranteed to be next larger palindrome. So, we must take the mirror of left side and copy it to right side. But there are some cases that must be handled in different ways. See the following steps.

We will start with two indices i and j. i pointing to the two middle elements (or pointing to two elements around the middle element in case of n being odd). We one by one move i and j away from each other.

Step 1. Initially, ignore the part of left side which is same as the corresponding part of right side. For example, if the number is "8 3 **4 2 2 4** 6 9″, we ignore the middle four elements. i now points to element 3 and j now points to element 6.

Step 2. After step 1, following cases arise:

Case 1: Indices i & j cross the boundary.
This case occurs when the input number is palindrome. In this case, we just add 1 to the middle digit (or digits in case n is even) propagate the carry towards MSB digit of left side and simultaneously copy mirror of the left side to the right side.
For example, if the given number is "1 2 9 2 1″, we increment 9 to 10 and propagate the carry. So the number becomes "1 3 0 3 1″

Case 2: There are digits left between left side and right side which are not same. So, we just mirror the left side to the right side & try to minimize the number formed to guarantee the next smallest palindrome.
In this case, there can be **two sub-cases**.

2.1) Copying the left side to the right side is sufficient, we don't need to increment any digits and the result is just mirror of left side. Following are some examples of this sub-case.
Next palindrome for "7 **8** 3 3 2 2″ is "7 8 3 3 8 7″
Next palindrome for "1 2 **5** 3 2 2″ is "1 2 5 5 2 1″
Next palindrome for "1 4 **5** 8 7 6 7 8 3 2 2″ is "1 4 5 8 7 6 7 8 5 4 1″
How do we check for this sub-case? All we need to check is the digit just after the ignored part in step 1. This digit is highlighted in above examples. If this digit is greater than the corresponding digit in right side digit, then copying the left side to the right side is sufficient and we don't need to do anything else.

2.2) Copying the left side to the right side is NOT sufficient. This happens when the above defined digit of left side is smaller. Following are some

examples of this case.
Next palindrome for “7 **1** 3 3 2 2” is “7 1 4 4 1 7”
Next palindrome for “1 2 **3** 4 6 2 8” is “1 2 3 5 3 2 1”
Next palindrome for “9 4 **1** 8 7 9 7 8 3 2 2” is “9 4 1 8 8 0 8 8 1 4 9”
We handle this subcase like Case 1. We just add 1 to the middle digit (or digits in ase n is even) propagate the carry towards MSB digit of left side and simultaneously copy mirror of the left side to the right side.

```
#include <stdio.h>

// A utility function to print an array
void printArray (int arr[], int n);

// A utility function to check if num has all 9s
int AreAll9s (int num[], int n );

// Returns next palindrome of a given number num[].
// This function is for input type 2 and 3
void generateNextPalindromeUtil (int num[], int n )
{
    // find the index of mid digit
    int mid = n/2;

    // A bool variable to check if copy of left side to
    bool leftsmaller = false;

    // end of left side is always 'mid -1'
    int i = mid - 1;

    // Begining of right side depends if n is odd or eve
    int j = (n % 2)? mid + 1 : mid;

   // Initially, ignore the middle same digits
    while (i >= 0 && num[i] == num[j])
        i--,j++;

    // Find if the middle digit(s) need to be incremente
    // side is not sufficient)
    if ( i < 0 || num[i] < num[j])
        leftsmaller = true;

    // Copy the mirror of left to tight
    while (i >= 0)
    {
        num[j] = num[i];
        j++;
        i--;
    }

    // Handle the case where middle digit(s) must be inc
    // This part of code is for CASE 1 and CASE 2.2
    if (leftsmaller == true)
    {
        int carry = 1;
        i = mid - 1;

        // If there are odd digits, then increment
        // the middle digit and store the carry
        if (n%2 == 1)
        {
            num[mid] += carry;
            carry = num[mid] / 10;
            num[mid] %= 10;
            j = mid + 1;
        }
        else
            j = mid;

        // Add 1 to the rightmost digit of the left side
        // towards MSB digit and simultaneously copying
        // to the right side.
        while (i >= 0)
        {
            num[i] += carry;
            carry = num[i] / 10;
            num[i] %= 10;
            num[j++] = num[i--]; // copy mirror to right
        }
    }
}

// The function that prints next palindrome of a given n
// with n digits.
void generateNextPalindrome( int num[], int n )
{
    int i;

    printf("\nNext palindrome is:\n");

    // Input type 1: All the digits are 9, simply o/p 1
    // followed by n-1 0's followed by 1.
    if( AreAll9s( num, n ) )
    {
        printf( "1 ");
        for( i = 1; i < n; i++ )
            printf( "0 " );
        printf( "1" );
    }

    // Input type 2 and 3
    else
    {
        generateNextPalindromeUtil ( num, n );

        // print the result
        printArray (num, n);
    }
}

// A utility function to check if num has all 9s
int AreAll9s( int* num, int n )
{
    int i;
    for( i = 0; i < n; ++i )
        if( num[i] != 9 )
            return 0;
    return 1;
}

/* Utility that prints out an array on a line */
void printArray(int arr[], int n)
{
    int i;
    for (i=0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver Program to test above function
int main()
{
    int num[] = {9, 4, 1, 8, 7, 9, 7, 8, 3, 2, 2};

    int n = sizeof (num)/ sizeof(num[0]);

    generateNextPalindrome( num, n );
    return 0;
}
```

Output:

```
Next palindrome is:
9 4 1 8 8 0 8 8 1 4 9
```