Given an integer, write a function that returns true if the given number is palindrome, else false. For example, 12321 is palindrome, but 1451 is not palindrome.

Let the given number be *num*. A simple method for this problem is to first reverse digits of *num*, then compare the reverse of *num* with *num*. If both are same, then return true, else false.

Following is an interesting method inspired from method#2 of this post. The idea is to create a copy of *num* and recursively pass the copy by reference, and pass *num* by value. In the recursive calls, divide *num* by 10 while moving down the recursion tree. While moving up the recursion tree, divide the copy by 10. When they meet in a function for which all child calls are over, the last digit of *num* will be ith digit from the beginning and the last digit of copy will be ith digit from the end.

```
// A recursive C++ program to check whether a given number is
// palindrome or not
#include <stdio.h>

// A function that reurns true only if num contains one digit
int oneDigit(int num)
{
    // comparison operation is faster than division operation.
    // So using following instead of "return num / 10 == 0;"
    return (num >= 0 && num < 10);
}

// A recursive function to find out whether num is palindrome
// or not. Initially, dupNum contains address of a copy of num.
bool isPalUtil(int num, int* dupNum)
{
    // Base case (needed for recursion termination): This statement
    // mainly compares the first digit with the last digit
    if (oneDigit(num))
        return (num == (*dupNum) % 10);

    // This is the key line in this method. Note that all recursive
    // calls have a separate copy of num, but they all share same copy
    // of *dupNum. We divide num while moving up the recursion tree
    if (!isPalUtil(num/10, dupNum))
        return false;

    // The following statements are executed when we move up the
    // recursion call tree
    *dupNum /= 10;

    // At this point, if num%10 contains i'th digit from beiginning,
    // then (*dupNum)%10 contains i'th digit from end
    return (num % 10 == (*dupNum) % 10);
}

// The main function that uses recursive function isPalUtil() to
// find out whether num is palindrome or not
int isPal(int num)
{
    // If num is negative, make it positive
    if (num < 0)
       num = -num;

    // Create a separate copy of num, so that modifications made
    // to address dupNum don't change the input number.
    int *dupNum = new int(num); // *dupNum = num

    return isPalUtil(num, dupNum);
}

// Driver program to test above functions
int main()
{
```

```
    int n = 12321;
    isPal(n)? printf("Yes\n"): printf("No\n");

    n = 12;
    isPal(n)? printf("Yes\n"): printf("No\n");

    n = 88;
    isPal(n)? printf("Yes\n"): printf("No\n");

    n = 8999;
    isPal(n)? printf("Yes\n"): printf("No\n");
    return 0;
}
```

Output:

```
Yes
No
Yes
No
```