Print all Jumping Numbers smaller than or equal to a given value

A number is called as a Jumping Number if all adjacent digits in it differ by 1. The difference between '9' and '0' is not considered as 1.
All single digit numbers are considered as Jumping Numbers. For example 7, 8987 and 4343456 are Jumping numbers but 796 and 89098 are not.

Given a positive number x, print all Jumping Numbers smaller than or equal to x. The numbers can be printed in any order.

Example:

```
Input: x = 20
Output:  0 1 2 3 4 5 6 7 8 9 10 12

Input: x = 105
Output:  0 1 2 3 4 5 6 7 8 9 10 12
      21 23 32 34 43 45 54 56 65
      67 76 78 87 89 98 101

Note: Order of output doesn't matter,
i,e., numbers can be printed in any order
```

We strongly recommend that you click here and practice it, before moving on to the solution.

One **Simple Solution** is to traverse all numbers from 0 to x. For every traversed number, check if it is a Jumping number. If yes, then print it. Otherwise ignore it. Time Complexity of this solution is O(x).

An **Efficient Solution** can solve this problem in O(k) time where k is number of Jumping Numbers smaller than or equal to x. The idea is use BFS or DFS.

Assume that we have a graph where the starting node is 0 and we need to traverse it from the start node to all the reachable nodes.

With the restrictions given in the graph about the jumping numbers, what do you think should be the restrictions defining the next transitions in the graph.

```
Lets take a example for input x = 90

Start node = 0
From 0, we can move to 1 2 3 4 5 6 7 8 9
[these are not in our range so we don't add it]

Now from 1, we can move to 12 and 10
From 2, 23 and 21
From 3, 34 and 32
.
.
.
.
.
.
and so on.
```

Below is BFS based C++ implementation of above idea.

```
// Finds and prints all jumping numbers smaller than or
// equal to x.
#include<bits/stdc++.h>
using namespace std;

// Prints all jumping numbers smaller than or equal to x starting
// with 'num'. It mainly does BFS starting from 'num'.
void bfs(int x, int num)
{
   // Create a queue and enqueue 'i' to it
   queue<int > q;
   q.push(num);

   // Do BFS starting from i
   while (!q.empty())
   {
      num = q.front();
      q.pop();

      if (num <= x)
      {
         cout << num << " ";
         int last_dig = num % 10;

         // If last digit is 0, append next digit only
         if (last_dig == 0)
            q.push((num*10) + (last_dig+1));


         // If last digit is 9, append previous digit only
         else if (last_dig == 9)
            q.push( (num*10) + (last_dig-1) );

         // If last digit is neighter 0 nor 9, append both
         // previous and next digits
         else
         {
            q.push((num*10) + (last_dig-1));
            q.push((num*10) + (last_dig+1));
         }
      }
   }
}

// Prints all jumping numbers smaller than or equal to
// a positive number x
void printJumping(int x)
{
   cout << 0 << " ";
   for (int i=1; i<=9 && i<=x; i++)
     bfs(x, i);
}

// Driver program
int main()
{
   int x = 40;
   printJumping(x);
   return 0;
}
```

Output:

```
0 1 10 12 2 21 23 3 32 34 4 5 6 7 8 9
```