PRIME DIGIT REPLACEMENT

```
#include <bits/stdc++.h>
using namespace std;

vector<vector<int>> primes;
set<int> S;
set<string> ans;

void Sieve(int n)
{
   primes.resize(8);
   vector<bool> sieve(11000100, false);
   int p = 2;
   sieve[2] = true;

   int digits = 1;
   int next = 10;

   while(p <= n)
   {
      if(p > next)
      {
         next *= 10;
         digits++;

         if(digits > 7) break;
      }
      primes[digits].push_back(p);
      S.insert(p);

      for(int i=2; p*i <= n; i++)
      {
```

```
            sieve[p*i] = true;
        }
        while(p < sieve.size() && sieve[p]) p++;
        sieve[p] = true;
    }
}

void Solve(string s, vector<int> &indices, vector<string> vis, int L)
{
    if(!ans.empty() && *ans.rbegin() < s) return;

    vis.push_back(s);

    set<string> total(vis.begin(), vis.end());

    if(total.size() == L)
    {
        if(ans.empty() || total < ans)
        {
            ans = total;
        }
        return;
    }

    char c = vis.back()[indices[0]]+1;

    for(; c <= '9'; c++)
    {
        string next = s;

        for(auto it : indices)
```

```
        {
            if(it == 0 && c == '0') continue;
            next[it] = c;
        }

        if(!ans.empty() && next > *ans.rbegin()) return;
        if(S.count(stoi(next)))
        {
            if(next <= s) return;
            Solve(next, indices, vis, L);
        }
    }
}


void NextCombination(vector<int> &indices)
{
    indices.back()--;

    for(int j=indices.size()-2; j>=0; j--)
    {
        if(indices[j] < indices[j+1]-1)
        {
            indices[j]++;
            j++;

            while(j < indices.size())
            {
                indices[j] = indices[j-1] + 1;
                j++;
            }
```

```
            return;
        }
    }
}


int main()
{
    Sieve(11000000);

    int n, k, l;
    cin >> n >> k >> l;

    vector<int> indices;

    for(int i=0; i<k; i++) indices.push_back(i);

    while(1)
    {
        if(indices.back() == n)
        {
            if(k == 1 || indices.front() == n - k) break;

            NextCombination(indices);
        }
        vector<int> temp = indices;

        for(int i=0; i<primes[n].size(); i++)
        {
            string s = to_string(primes[n][i]);
```

```
            if(!ans.empty() && s > *ans.begin()) break;

            bool valid = true;

            for(auto it : temp)
            {
               if(k != n && s[it] != s[temp[0]])
               {
                  valid = false;
                  break;
               }
            }
            if(!valid) continue;

            Solve(s, temp, {}, l);
         }
         indices.back()++;
      }
      for(auto it : ans)
      {
         cout << it << ' ';
      }
      cout << endl;

      return 0;
}
```