[redacted] [redacted] [redacted]

240-HW-8
4 Hours.
130527

```
// Written [redacted]
//
// solves Train Shuffle Problem for Homework 8.
//
import java.util.LinkedList;
import java.util.Queue;
import java.util.Arrays;
////////////////////////////////////////////////////////////////////////
class Shuffle
{
//--------------------------------------------------------------------
   public static void main(String[] args)
   {
      if ( args.length != 2 )
      {
         System.out.println("Usage: java Shuffle <tracks> " +
                            "<permutation>");
         return;
      }

      if ( !isValid(args[1]) )
      {
         System.out.println("Invalid Permutation");
         return;
      }

      Queue<Integer>[] track = new Queue[Integer.parseInt(args[0])];
      for ( int i = 0; i < track.length; i++ )
         track[i] = new LinkedList<Integer>();

      int expectedCar = 1;

      for ( char c : args[1].toCharArray() )
      {
         int inputCar = Character.digit( c, 10 );

         if ( inputCar == expectedCar )
         {
            System.out.printf("Move %d directly to output " +
                              "using track %d\n",
                              inputCar, track.length - 1);
            expectedCar++;
            continue;
         }

         int movedToTrack = getMovedToTrack( inputCar, track );
         if ( movedToTrack >= 0 )
         {
            System.out.printf("Move %d to track %d\n",
                               inputCar, movedToTrack);

            track[movedToTrack].add(inputCar);
            continue;
         }

         int emptyTrack = getEmptyTrack( track );
         if ( emptyTrack >= 0 )
```

```
            {
               System.out.printf("Move %d to empty track %d\n",
                                 inputCar, emptyTrack);

               track[emptyTrack].add(inputCar);
               continue;
            }

            System.out.println("No solution");
            return;
         }

         int movedFromTrack = getMovedFromTrack( expectedCar, track );
         while( movedFromTrack >= 0 )
         {
            System.out.printf("Move %d from track %d to output\n",
                              expectedCar, movedFromTrack);
            expectedCar++;
            movedFromTrack = getMovedFromTrack( expectedCar,track );
         }
      }
   //-----------------------------------------------------------------------
      public static int getMovedToTrack(int car, Queue<Integer>[] track)
      {
         for ( int i = 0; i < track.length - 1; i++ )
         {
            int[] a = new int[track[i].size()];

            for ( int j = 0; j < a.length; j++ )
            {
               a[j] = track[i].remove();
               track[i].add(a[j]);
            }

            if ( a.length > 0 && a[a.length-1] < car  )
               return i;
         }

         return -1;
      }
   //-----------------------------------------------------------------------
      public static int getMovedFromTrack(int car,Queue<Integer>[] track)
      {
         for ( int i = 0; i < track.length - 1; i++ )
         {
            if ( !track[i].isEmpty() && track[i].peek() == car )
            {
               track[i].remove();
               return i;
            }
         }

         return -1;
      }
   //-----------------------------------------------------------------------
      public static int getEmptyTrack(Queue<Integer>[] track)
      {
```

```
      for ( int i = 0; i < track.length - 1; i++ )
         if ( track[i].isEmpty() ) return i;

      return -1;
   }
//-----------------------------------------------------------------------
   public static boolean isValid(String input)
   {
      int[] num = new int[input.length()];
      for ( int i = 0; i < num.length; i++ )
         num[i] = Character.digit( input.charAt(i), 10 );

      Arrays.sort(num);

      if ( num[0] != 1 ) return false;

      for ( int i = 0; i < num.length - 1; i++ )
         if ( num[i+1] != num[i] + 1 ) return false;

      return true;
   }
//-----------------------------------------------------------------------
}// end class Shuffle
/////////////////////////////////////////////////////////////////////////
```

[redacted] [redacted] [redacted]

240-HW-08

3 hrs

130527

```
// Written b[redacted]
//
// Solves Spring 2013 CS 240 Homework 08.
//
import java.io.*;
import java.util.Queue;
import java.util.LinkedList;
////////////////////////////////////////////////////////////////////////
class Shuffle
{
//-----------------------------------------------------------------------
   public static void main ( String [] args ) throws Exception
   {
      if(args.length != 2)
      {
         System.out.println(
               "Usage: java Shuffle <tracks> <permutation>");
         return;
      }
      Queue<Integer>[] tracks = new Queue[Integer.parseInt(args[0])];
      for(int i=0; i<tracks.length; i++)
         tracks[i] = new LinkedList<Integer>();
      char[] init = args[1].toCharArray();
      int outNeeded = 1;
      int counter = 0;
      boolean noSolution = false;
      while(true)
      {
         if((counter < init.length)&&
               (outNeeded == Character.digit(init[counter],10)))//rule 1
         {
            System.out.println("Move " + outNeeded +
                  " directly to output using track " +
                  (tracks.length-1));
            counter++;
            outNeeded++;
            continue;
         }
         for(int i=0; i<tracks.length; i++)//rule 2
         {
            if((tracks[i].size() != 0)&&
                  (tracks[i].peek() == outNeeded))//rule 2
            {
               System.out.println("Move " + tracks[i].remove() +
                     " from track " + i + " to output");
               outNeeded++;
            }
         }
         if(counter<init.length)
         {
            for(int i=0; i<tracks.length; i++)//rule 3-5
            {
               if(i == tracks.length - 1)//rule 5
               {
                  System.out.println("No solution");
                  noSolution = true;
               }
               else if(tracks[i].size() == 0)//rule 4
               {
                  tracks[i].add(Character.digit(init[counter], 10));
                  System.out.println("Move " +
```

```
                        Character.digit(init[counter], 10) +
                        " to empty track " + i);
                    counter++;
                    break;
                }
                else if(((LinkedList<Integer>)tracks[i]).getLast() <
                      Character.digit(init[counter], 10))//rule 3
                {
                    tracks[i].add(Character.digit(init[counter], 10));
                    System.out.println("Move " +
                        Character.digit(init[counter], 10) +
                        " to track " + i);
                    counter++;
                    break;
                }
            }
        }
        if((noSolution)||
            ((counter >= init.length)&&(outNeeded > init.length)))
        {
            break;
        }
    }
  }
//---------------------------------------------------------------------------
} // end class Shuffle
////////////////////////////////////////////////////////////////////////////
```