```
#!/usr/bin/perl -w
# Author: lh3
# Note: Ideally, this script should be written in C. It is a bit slow at present.
use strict;
use warnings;
use Getopt::Std;
my %opts;
my $version = '0.1.1';
my $usage = qq{
Usage: solid2fastq.pl <in.title> <out.prefix>

Note: <in.title> is the string showed in the `# Title:' line of a
      ".csfasta" read file. Then <in.title>F3.csfasta is read sequence
      file and <in.title>F3_QV.qual is the quality file. If
      <in.title>R3.csfasta is present, this script assumes reads are
      paired; otherwise reads will be regarded as single-end.

      The read name will be <out.prefix>:panel_x_y/[12] with `1' for F3
      tag and `2' for R3. Usually you may want to use short <out.prefix>
      to save diskspace. Long <out.prefix> also causes troubles to maq.

};

getopts('', \%opts);
die($usage) if (@ARGV != 2);
my ($title, $pre) = @ARGV;
my (@fhr, @fhw);
my @fn_suff = ('F3.csfasta', 'F3_QV.qual', 'R3.csfasta', 'R3_QV.qual');
my $is_paired = (-f "$title$fn_suff[2]" || -f "$title$fn_suff[2].gz")? 1 : 0;
if ($is_paired) { # paired end
  for (0 .. 3) {
        my $fn = "$title$fn_suff[$_]";
        $fn = "gzip -dc $fn.gz |" if (!-f $fn && -f "$fn.gz");
        open($fhr[$_], $fn) || die("** Fail to open '$fn'.\n");
  }
  open($fhw[0], "|gzip >$pre.read1.fastq.gz") || die;
  open($fhw[1], "|gzip >$pre.read2.fastq.gz") || die;
  open($fhw[2], "|gzip >$pre.single.fastq.gz") || die;
  my (@df, @dr);
  @df = &read1(1); @dr = &read1(2);
  while (@df && @dr) {
        if ($df[0] eq $dr[0]) { # mate pair
          print {$fhw[0]} $df[1]; print {$fhw[1]} $dr[1];
          @df = &read1(1); @dr = &read1(2);
        } else {
          if ($df[0] le $dr[0]) {
                print {$fhw[2]} $df[1];
                @df = &read1(1);
          } else {
                print {$fhw[2]} $dr[1];
                @dr = &read1(2);
          }
        }
  }
  if (@df) {
	print {$fhw[2]} $df[1];
	while (@df = &read1(1, $fhr[0], $fhr[1])) {
	  print {$fhw[2]} $df[1];
	}
  }
  if (@dr) {
	print {$fhw[2]} $dr[1];
	while (@dr = &read1(2, $fhr[2], $fhr[3])) {
	  print {$fhw[2]} $dr[1];
	}
  }
  close($fhr[$_]) for (0 .. $#fhr);
  close($fhw[$_]) for (0 .. $#fhw);
} else { # single end
  for (0 .. 1) {
	my $fn = "$title$fn_suff[$_]";
	$fn = "gzip -dc $fn.gz |" if (!-f $fn && -f "$fn.gz");
	open($fhr[$_], $fn) || die("** Fail to open '$fn'.\n");
  }
  open($fhw[2], "|gzip >$pre.single.fastq.gz") || die;
  my @df;
  while (@df = &read1(1, $fhr[0], $fhr[1])) {
	print {$fhw[2]} $df[1];
  }
  close($fhr[$_]) for (0 .. $#fhr);
  close($fhw[2]);
}

sub read1 {
  my $i = shift(@_);
  my $j = ($i-1)<<1;
  my ($key, $seq);
  my ($fhs, $fhq) = ($fhr[$j], $fhr[$j|1]);
  while (<$fhs>) {
	my $t = <$fhq>;
	if (/^>(\d+)_(\d+)_(\d+)_[FR]3/) {
	  $key = sprintf("%.4d_%.4d_%.4d", $1, $2, $3); # this line could be improved on
64-bit machines
	  die(qq/** unmatched read name: '$_' != '$_'\n/) unless ($_ eq $t);
	  my $name = "$pre:$1_$2_$3/$i";
	  $_ = substr(<$fhs>, 2);
	  tr/0123./ACGTN/;
	  my $s = $_;
	  $_ = <$fhq>;
	  s/^(\d+)\s*//;
	  s/(\d+)\s*/chr($1+33)/eg;
	  $seq = qq/\@$name\n$s+\n$_\n/;
	  last;
	}
  }
  return defined($seq)? ($key, $seq) : ();
}
```