Fencing Tournament Structuring Problem

The goals of this programming exercise are to test your ability to handle TSV formatted files, create useful data structures, and design a useful algorithm to solve an unfamiliar problem (most likely unfamiliar!).

Create a Python program that can handle some of the organization required for running a fencing tournament. The code will take as input a file containing a list of tournament participants of various skill levels. The program will output a list of groups of fencers (called pools) that are equally weighted in difficulty. There are specific rules about how the fencers can be grouped (explained later). Create a flexible program that can handle any number of participants between 12 and 100 and create a correctly balanced list of pools. We will provide a few sample input files. Some data sets actually have several correct answers so be prepared to defend your answers.

Input files have the following information laid out in csv format:
LICHTEN, Keith, EBFG ,A15
TEPEDELENLIOGLU, Mehmet, EBFG ,A15
… #followed by the rest of the entries

The number of entrants will be between 12 and 100 long (inclusive). It is possible to not have a club affiliation in which case the club field will be empty. The rankings sorted in order from strongest to weakest are: A<year>, B<Year>, C<Year>, D<Year>, E<Year>, U. Letters are given the first order ascending sort followed by a secondary descending sort of the year. The more recent the year, the higher the ranking. So an A15 > A14 > A13 > B15 and so on.

Rules for the distribution of fencers into pools:
Arrange the fencers into groups (pools) of fencers to create a fair competition such that:

1. Size(largestPool) – Size(smallestPool) <= 1
2. Optimum size of the pools should be a mix of 6 and 7 fencers. If that cannot be achieved, then it should be a mix of 7 and 8 fencers. If that cannot be achieved, then it should be a mix of 5 and 6 fencers. (see note for #2 below)
3. Distribute members from the same club evenly throughout the pools so that they cannot collude for club benefit (see note for #3 below)
4. Distribute the fencers according to rank amongst the pools such that each pool has the same level of difficulty (as much as possible). Hint: a sorted list can be apportioned out to the pools in a serpentine pattern. So you would assign fencers to pools in an order like this for 3 pools: 1 2 3 3 2 1 1 2 3 3 2 1…

Example pool sizes for #2 above
L = 12 should have two pools of 6
L = 13 should have 1 pool of 6 and 1 pool of 7
L = 16 should have two pools of 8
L = 17 should have 1 pool of 5, and 2 pools of 6

Note for #3:
Members from the same club should be distributed to separate pools as much as possible. So if you have participants from the same club in the tournament they cannot be in the same pool unless there is no other choice. If you have 2 members from club HMC and only two pools, then they must be placed into separate pools. If you have 4 members from the same club and only 2 pools, then you will need to place 2 members into each pool. To resolve these sorts of conflicts with pool assignments follow these rules:

1. Resolve the conflict by swapping members from other pools of equal rank with one of the fencers in conflict
2. If you cannot use the above rule, then exchange the lower ranking club mate with the closest ranking fencer from another pool that is ranked below him.

Note: it is possible for a fencer to be unaffiliated to a club. It is not necessary to balance the numbers of unaffiliated fencers in the pools.

Your output should list the tournament information first followed by the member list from each pool, clearly labeled. You can print the results to stdout. You are welcome to use standard python libraries. Please use Python 2.7.

Example:

```
Competitor List
Keith          LICHTEN              EBFG            A      15
Alexandre         RACHTCHININE      FAIRFAX FENCERS CLUB         A      15
Mehmet            TEPEDELENLIOGLU           EBFG            A      15
David          HITCHCOCK            OLYMPIA F. C.          A      15
Sergey          SUPONYA             MEDEO F.C.             A      15
Halim          ALJIBURY             NO FEAR                A      14
Velizar        ILIEV          OLYMPIANFC            A      14
Tomas           STRAKA              SSC             A      12
Daniel         KROGH                NWFC                   B      15
Kenneth          HAGAN              LOUISVLLE F.C.         B      14
Alfred         ROEBUCK              NO FEAR                B      13
John           KISSINGFORD          NO FEAR                C      15
Aaron           PAGE          METRO TACOMA          C      15

Pool List
--)------- Pool # 1 -------(-- (7)
Keith          LICHTEN              EBFG            A      15
David          HITCHCOCK            OLYMPIA F. C.          A      15
Sergey          SUPONYA             MEDEO F.C.             A      15
Tomas           STRAKA              SSC             A      12
Daniel         KROGH                NWFC                   B      15
John           KISSINGFORD          NO FEAR                C      15
Aaron           PAGE          METRO TACOMA          C      15
```

```
--)------- Pool # 2 -------(-- (6)
Alexandre        RACHTCHININE        FAIRFAX FENCERS CLUB        A        15
Mehmet           TEPEDELENLIOGLU              EBFG            A        15
Halim          ALJIBURY                NO FEAR                A        14
Velizar        ILIEV            OLYMPIANFC            A        14
Kenneth          HAGAN                 LOUISVLLE F.C.         B        14
Alfred         ROEBUCK                 NO FEAR                B        13
```

Please send us your solution. Full functionality is not necessarily expected, but a description of why some parts were not implemented should accompany your solution if that is the case. Please present your code with all your usual readability and maintainability practices in place.

Hints:
There are three basic problems to deal with here.

1. Input processing – There are a number of little problems with the input data. Do not modify the input files to clean up these inconsistencies, handle the exceptions in the code.
2. Distribute the fencers amongst the pools such that each pool has the same number of A's, B's, etc (or as close as possible). See the example above.
3. Balance the distribution of the teams. The count for any club should be only +-1 from the count for the same club in any other pool. So it is an error if you have 3 EBFG club members in one pool and only one EBFG member in another pool. This rule applies to all clubs.

Basic level of implementation: able to ingest data and produce a list that is sorted correctly (as explained above)

Intermediate level of implementation: Output of pools with balanced distribution of ranking (balancing the pools based on letter/year rankings). This should also be able to handle a variety of tournament sizes from 12 to 100, and create the correct pool sizes.

Expert level of implementation: Output of pools with balanced distribution of ranking (as above) and team affiliation