EE2331 Data Structures and Algorithms

Assignment 1

You are a member of the SW team developing a program to store records of CityU long jump competition. Your job is to implement a data structure, called *record*, to store jumper data. Each record holds the jumper name, id, and list of results. Results are measured as centimeters, e.g. an integer value 200 means 200cm. Each competitor has unlimited number of jumps, thus you must use a linked list for storing the results. Results are stored oldest first and new results can be added to the end of list. There is also a team competition. A jumper record is used to present an individual or a team of two jumpers. A team is formed by combining two individual jumpers (using *createTeam* function).

Specific rules applie for the team formation:

- Jumpers are ranked first according to their max result. Jumpers with the same max result are ranked according to their average result (sum of all results)/(number of attempts). If averages results are the same, rankings are considered to be the same.
- Team name and team id are formed my concatenating the two jumper names or ids. Such that the jumper with higher ranking will get his/her name/id presented first. E.g. *name_higher_rank* + *name_lower_rank* and *id_higher_rank* + *id_lower_rank*. If the rankings are the same, the order is as follows: *this.name* + *other.name* and *this.id* + *other.id*.
- When two jumpers form a team, the initial team results are composed based on individual results such that for each attempt the better individual result will be counted. E.g. jumper1 [100,90,200,400] and jumper2 [50,190] result team [100,190,200,400]. Missing attempts are interpreted as 0.

Three program files are given to you in this exercise.

1. You should not make any changes to *linkedListType.h*
2. You can modify the main function and add additional test cases in *assignment_1.cpp*
3. Implement the member and non-member functions for the class **record** specified in *record.h*. DO NOT modify the function signatures given to you but you can define additional **private** member functions as you need. The function specifications are written in their comment section.
   - `record(const string& id_, const string& name_, const int& init_result);`
   - `void addResult(int result);`
   - `record& createTeam(record& other);`
   - `int compare(record& other);`
   - `friend ostream& operator<<(ostream& os, record& record);`
4. You ONLY need to submit the file *record.h* to Canvas. Write your name and student ID in the comment section at the top.