# Data Structure Project 6

# Quick Sort and Heap Sort

# Introduction

Baseball statistics play an important role in summarizing players' performance. Design a program to sort records of players in a table using **quick sort** and **heap sort**.

# Requirements

## Project requirements

Store a table which contains the performance of all players in your program. For each player, the following statistics are recorded:

- *string name*;
- *int ab*; //a positive integer, always greater than zero
- *int h*; //a non-negative integer
- *int hr*; //a non-negative integer
- *int tb*; //a non-negative integer
- *int bb*; //a non-negative integer
- *double avg*; //avg = (h/ab)
- *double obp*; // obp = (h+bb)/(ab+bb)
- *double slg*; // slg = (tb/ab)
- *double ops*; // ops = (obp + slg)

Table 1 is an example of a set of possible data:

| name | ab | h | hr | tb | bb | avg | obp | slg | ops |
|---|---|---|---|---|---|---|---|---|---|
| Suzuki | 227 | 73 | 5 | 103 | 5 | 0.322 | 0.336 | 0.454 | 0.790 |
| Jeter | 683 | 216 | 15 | 293 | 45 | 0.316 | 0.358 | 0.429 | 0.787 |
| Cano | 627 | 196 | 33 | 345 | 61 | 0.313 | 0.373 | 0.550 | 0.924 |
| Granderson | 596 | 138 | 43 | 293 | 75 | 0.232 | 0.317 | 0.492 | 0.809 |

Table 1

Write a program implementing quick sort and heap sort algorithms. Each time you have to sort the table **twice** according to two different attributes. That is, if two or more entries (players) have the same value, sort them again using another designated attribute. Use **quick sort** for the first round and **heap sort** for the second round. The size of the table is at most 100 entries.

## Program requirements

- Standard C++ language, without any external or OS-dependent library
  - Visual C++ 2010 or GCC 4.5/4.6 are recommended
- C++ STL Containers (vector, deque, list, stack, queue, …) and STL Sorting Algorithm are not allowed
- Do not include any system command functions (for example, *system("pause")*; ) in your program

# Input and Output

## Input

The input data will be stored in a text file, one operation each line. For one line, the first character indicates an operation, followed by some arguments if necessary.

- ***I***: insert an entry to the table. This operation contains six arguments, separated by a space character between two arguments. The content of the argument is *(name ab h hr tb bb)*, where *name* is a string indicating the name of the player, *ab* is a positive integer, and other arguments are all non-negative integers. After reading the line, you should establish an entry of the player and find the *avg, obp, slg,*

*ops* values respectively, then insert the data into a table (array / list).

- ***S***: sort the table according to the designated attributes. This operation has two different arguments, indicating the attributes to sort. The attributes are represented as following:
  - ***AB***: sort by *ab* in descending order.
  - ***H***: sort by *h* in descending order.
  - ***HR***: sort by *hr* in descending order.
  - ***TB***: sort by *tb* in descending order.
  - ***BB***: sort by *bb* in descending order.
  - ***AVG***: sort by *avg* in descending order.
  - ***OBP***: sort by *obp* in descending order.
  - ***SLG***: sort by *slg* in descending order.
  - ***OPS***: sort by *ops* in descending order.

  For example, an operation may be "*S AB BB*" which means sorts the table by *ab* values using quick sort, and if two or more entries (players) have the same *ab* value, sort them by *bb* values using heap sort. If both *ab* and *bb* values are the same, any kind of ordering is allowed.

  After sorting, print the result in *(name, attribute 1, attribute 2);* sequence format. Put a space character after each comma and semicolon. When printing the *avg, obp, slg, ops* values, set the precision to 3.
- ***C***: Clear the table. Remove all entries in the table.
- ***E***: End. Terminate your program.

**The filename of the input file will be passed through command line argument.** You should use the filename string passed to your main function.

## Output

The output format is described above in the ***S*** operation. Output the result to *stdout* using *std::cout* or *printf*.

## Example

| | |
|---|---|
| Input.txt | I Cabrera 622 205 44 377 66<br>S AB BB<br>I Granderson 596 138 43 293 75<br>I Hamilton 562 160 43 324 60<br>S HR H<br>I Jeter 683 216 15 293 45<br>S AVG HR<br>C<br>I Posey 530 178 24 291 69<br>I Suzuki 629 178 9 103 22<br>I Wright 581 178 21 286 81<br>S H AB<br>S H HR<br>S AVG H<br>E |
| Command arguments | Input.txt |
| Output | (Cabrera, 622, 66);<br>(Cabrera, 44, 205); (Hamilton, 43, 160); (Granderson, 43, 138);<br>(Cabrera, 0.330, 44); (Jeter, 0.316, 15); (Hamilton, 0.285, 43); (Granderson, 0.232, 43);<br>(Suzuki, 178, 629); (Wright, 178, 581); (Posey, 178, 530);<br>(Posey, 178, 24); (Wright, 178, 21); (Suzuki, 178, 9);<br>(Posey, 0.336, 178); (Wright, 0.306, 178); (Suzuki, 0.283, 178); |

Note: your program should output **exactly the same format** described above. Additional letters, punctuations, spaces, or characters will be treated as wrong results.

# Scoring Criteria

- Correctness 80%
  - TA will test your program with some basic data
- Extreme cases 10%
  - More difficult input data
- Source code quality and readability 10%
  - Comments, variable/function naming, consistency, …

# Project Submission

Please pack all your **source code** files (*.cpp and/or *.h) in the studentID.zip file, ex: *0016789.zip*, and upload to e3. DO NOT UPLOAD/CONTAIN OTHER FILES. The dead line is 23:59, 8 Jan. 2013.

Specify the compiler you used in the very beginning (first/second line) of your codes using comments is recommended.

# Note

Plagiarism / copy others work are not allowed.

If you have any problem about this project, please contact TA through email or come to EC 637 for more information.