**CPP Problem Design Example**

|  |
| --- |
| **Subject: Report analyze** |
| **Contributor: 鍾賢廣** |
| **Main testing concept: File In**   |  |  | | --- | --- | | **Basics** | **Functions** | | ■ C++ BASICS  ■ FLOW OF CONTROL  ■ FUNCTION BASICS  □ PARAMETERS AND OVERLOADING  ■ ARRAYS  □ STRUCTURES AND CLASSES  □ CONSTRUCTORS AND OTHER TOOLS  □ OPERATOR OVERLOADING, FRIENDS, AND REFERENCES  ■ STRINGS  □ POINTERS AND DYNAMIC ARRAYS | □ SEPARATE COMPILATION AND NAMESPACES  □ STREAMS AND FILE I/O  □ RECURSION  □ INHERITANCE  □ POLYMORPHISM AND VIRTUAL FUNCTIONS  □ TEMPLATES  □ LINKED DATA STRUCTURES  □ EXCEPTION HANDLING  □ STANDARD TEMPLATE LIBRARY  □ PATTERNS AND UML | |
| **Description:**  Write a data analyzing program that reads data from the *INPUT* file and outputs the result for  each command. In the *INPUT* file, each line consists of the *STUDENT\_INFO*, which begins  with the *STUDENT\_NAME* and is followed by the *COURSE\_INFO*s of the courses that the  student takes. All information is separated by a comma ','. Below is the detail format of the  information:  *STUDENT\_NAME*:  Only English letters and the space character ' ' are allowed and there should be no  more than 256 characters.  *COURSE\_INFO*:  Consists of the *COURSE\_ID* and the *COURSE\_GRADE*, and they are separated  by a colon ':'.  *COURSE\_ID*:  Expressed as a 3-digit non-negative number and would not be greater than  1000.  *COURSE\_GRADE*:  A non-negative integer and won’t exceed 100.  Students will not participate in more than 256 courses. Also, we assume the student name is identical. If duplicated name is discovered, keep the information of the first student and discard the others. Here is an example of student data:  Obama Hank,001:6,045:89,999:100  The analyzing program also reads the command input from command line. Each command has  its own parameters and should all be separated by the space character ' '. The detail descriptions  of each command are listed below:  *STUDENT* *STUDENT\_NAME*:  Outputs the *COURSE\_INFO*s that *STUDENT\_NAME* takes and also the average  score of the student. The *COURSE\_INFO*s should be ordered by *COURSE\_ID* in  ascending order. The output should follow the format of the following example:  name:Obama Hank  002:45  012:55  365:35  average:50  If there is no student named *STUDENT\_NAME*  then output “Student not found”  instead.  *COURSE COURSE\_ID*:  Outputs the scores of all students who participate in *COURSE\_ID* and  the average score of the course. The student scores should be ordered by  *STUDENT\_NAME* in ascending order(alphabetical order). The output should  follow the format of the following example:  course:001  Michael Jackson:12  Obama Hank:3  Tom Cruise:65  average:26.6667  *RANK*:  Outputs all the student’s average scores in descending order. For those who have  the same average scores, the output should be sorted by their names in ascending  order(alphabetical order). Here is an example output of *RANK*:  Obama Hank:33  Tom Cruise:26.6667  Michael Jackson:5.25  *ADD STUDENT\_INFO*:  Adds student information manually. Here is an example of the input command:  ADD Otako Kanawa,033:0,052:89,999:100  If the student’s information is already existed, replace the information with the  new one.  **Input:**  The program should read data from the file “record.txt”. The file format can be referred to the  “example.txt” file.  The data file is assumed to be placed at the same directory of the compiled program.  **Output:**  **The *COURSE\_ID* should be outputted as a 3-digit non-negative integer number.**  The floating point values will be output according to default output settings namely you did not need to change the output settings. Please refer to the sample output below.  **Sample Input / Output：**   |  |  | | --- | --- | | Sample Input | Sample Output | | \*/read “example.txt”/\*  STUDENT aHellowWorld  STUDENT Amaan Prince  STUDENT Warren Mills  RANK  COURSE 999  COURSE 100  COURSE 080  ADD Warren Mills,069:0,054:0,678:0,999:89,080:100  ADD AHellowWorld,001:25,015:65,521:78,999:100  COURSE 001  COURSE 080  STUDENT aHellowWorld  RANK | STUDENT aHellowWorld  No Student  STUDENT Amaan Prince  name:Amaan Prince  001:25  015:65  521:78  999:100  average:67  STUDENT Warren Mills  name:Warren Mills  054:0  069:0  678:0  999:89  average:22.25  RANK  Khalid Scott:89  Amaan Prince:67  Todd Baker:55.25  George Raymond:50  Saul Benjamin:39  Jesse Orr:38.25  Maximilian Perez:33.6667  Harold Rosales:29.6667  Warren Mills:22.25  Adil Garrison:0  COURSE 999  course:999  Amaan Prince:100  Jesse Orr:23  Khalid Scott:56  Saul Benjamin:56  Todd Baker:85  Warren Mills:89  average:68.1667  COURSE 100  course:100  Harold Rosales:33  Maximilian Perez:33  Saul Benjamin:100  average:55.3333  COURSE 080  course:80  No course record  ADD Warren Mills,069:0,054:0,678:0,999:89,080:100  ADD AHellowWorld,001:25,015:65,521:78,999:100  COURSE 001  course:1  AHellowWorld:25  Amaan Prince:25  George Raymond:100  average:50  COURSE 080  course:80  Warren Mills:100  average:100  STUDENT aHellowWorld  No Student  RANK  Khalid Scott:89  AHellowWorld:67  Amaan Prince:67  Todd Baker:55.25  George Raymond:50  Saul Benjamin:39  Jesse Orr:38.25  Warren Mills:37.8  Maximilian Perez:33.6667  Harold Rosales:29.6667  Adil Garrison:0 | |
| **■ Easy, Only basic programming syntax and structure are required.**  **□ Medium, Multiple programming grammars, and structures are required.**  **□ Hard, Need to use multiple program structures or complex data types.** |
| **Expected solving time:**  10 minutes |
| **Other notes:**  The Program should take multiple test input, Finish when reading EOF.  alphabetical order: A<Z<a<z.  The floating point values will be output according to default output settings namely you did not need to change the output settings. |