**11002 CPP Midterm Exam**

|  |
| --- |
| **Subject: Grade Recorder** |
| **Contributor: 柯鐘奇** |
| **Main testing concept: Class**   |  |  | | --- | --- | | **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 program that can record students’ grades. There are two class functions (**Student**, **Class**) that you should implement in your program. The class “Student” contains operations of defining a student and their grades, and the class **Class** contains operations of the whole class. Beware that you should write your two class functions under the given “Recorder.cpp” and define these functions under “Recorder.h”, as we will use our own main.cpp to call “Recorder.h”.  […] in the description should be replaced.  The class **Student** should include the following member functions:  Note: Student should have their ID number from 0.  Note: The name could be pass or default. The default name is Student[count]. If the pass name is invalided (exist name, “Student” prefix…), you should use default name. The ID and the [count] are different. The [count] should be started from 0.   * **Constructor** * **Student();**   If no student name is input, you should name it with the format “Student[count]”.  e.g. Student0, Student1…   * **Student(std::string name);**   If the input name exists, rename it with "Student[count]" and print  “Error: The name [name] exists. Use default name: Student[count].”.  If the input name has “Student” as prefix, rename it with "Student[count]" and print  “Error: The name Student is reserved word. Use default name: Student[count].”.  e.g. Student, Studentke, Studentlai, Student123…(illegal)  Beware that it is case sensitive.   * **void addGrade(std::string subject, int point);**   Record grade into student data.  If the subject exists, replace the grade with the new one.   * **void deleteGrade(std::string subject);**   Delete the grade.  If the subject doesn’t exist, print “Error: Subject [subject] could not be found.”.   * **void deleteGrade();**   Delete all the grades.   * **void printGrade() ;**   Print out the student’s ID, name, subject and grade as the form shown below.  Example:   |  | | --- | | ID: 0 Name: Student0  Chinese 87  OOP 60 |   If no any grade exists, print “Error: Grade could not be found.”.  The class **Class** should include the following member functions:   * **Constructor** * **Class();** * **void addStudent(Student student);**   Add new student to *Class*. If the student is already in class, print  “Error: The student [student.name] is already in the class.”.     * **void deleteStudent(Student student);**   Delete student from Class.  If student could not be found, print “Error: Student [student.name] could not be found.”.   * **void deleteStudent(std::string studentName);**   Delete student from Class.  If studentName could not be found, print  “Error: Student [studentName] could not be found.”.   * **void printNumOfStudent(std::string subjectName);**   Print out the subject name and the number of students.  Example:   |  | | --- | | OOP 4 |   If subjectName could not be found, print  “Error: Subject [subjectName] could not be found.”.   * **void printNumOfStudent();**   Print out all the subject names and the number of students. You should consider the order of the student first. Then the order of the subject of that student.   * **void printAvgScore(std::string subjectName);**   Print out the subject name and its average score.  Example:   |  | | --- | | OOP 60 |   If subjectName could not be found, print  “Error: Subject [subjectName] could not be found.”   * **void printAvgScore( );**   Print out all the subject names and the average score of subjects. You should consider the order of the student first. Then the order of the subject of that student.   * **void printSubjectInformation(std::string subjectName);**   Print the subject's information in the order you added them.  Example:   |  | | --- | | Subject name: OOP  Num of students: 3  Student0 60  John 60  Student1 60 |   If subjectName could not be found, print “Error: Subject [subjectName] could not be found.”   * **void printSubjectInformation ( );**   Print out all the subject information. You should consider the order of the student first. Then the order of the subject of that student.  **Input:**  Substitution of your main function.  **Output:**  Please refer to the sample output.  **Sample Input / Output：**   |  |  | | --- | --- | | Sample Input | Sample Output | | According to the given main function | Error: The name John exists. Use default name: Student1.  Error: The name Student is reserved word. Use default name: Student2.  ID: 0 Name: Student0  OOP 60  Error: Subject OOP could not be found.  ID: 1 Name: John  Chinese 70  OOP 64  ID: 2 Name: Student1  OOP 70  English 60  Error: The student Student2 is already in the class.  Error: Student Student22 could not be found.  Chinese 1  OOP 2  English 1  Error: Subject oop could not be found.  OOP 2  Chinese 70  OOP 67  English 60  Error: Subject eng could not be found.  English 60  Subject name: Chinese  Num of students: 1  John 70  Subject name: OOP  Num of students: 2  John 64  Student1 70  Subject name: English  Num of students: 1  Student1 60  Error: Subject chin could not be found.  Subject name: Chinese  Num of students: 1  John 70 | |
| **□ 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:**60 minutes |
| **Main function:**  **#include "Recorder.h"**  **int main()**  **{**  **Student a = Student();**  **Student b = Student("John");**  **Student c = Student("John");**  **Student d = Student("Student77");**  **a.addGrade("OOP", 50);**  **a.addGrade("OOP", 60);**  **a.printGrade();**  **b.addGrade("OOP", 60);**  **b.deleteGrade();**  **b.deleteGrade("OOP");**  **b.addGrade("Chinese", 70);**  **b.addGrade("OOP", 64);**  **b.printGrade();**  **c.addGrade("OOP", 70);**  **c.addGrade("English", 60);**  **c.printGrade();**  **d.addGrade("OOP", 100);**  **Class cls;**  **cls.addStudent(a);**  **cls.addStudent(b);**  **cls.addStudent(c);**  **cls.addStudent(d);**  **cls.addStudent(d);**  **cls.deleteStudent(a);**  **cls.deleteStudent("Student2");**  **cls.deleteStudent("Student22");**  **cls.printNumOfStudent();**  **cls.printNumOfStudent("oop"); …ok**  **cls.printNumOfStudent("OOP"); …ok**  **cls.printAvgScore();**  **cls.printAvgScore("eng");**  **cls.printAvgScore("English");**  **cls.printSubjectInformation();**  **cls.printSubjectInformation("chin");**  **cls.printSubjectInformation("Chinese");**  **return 0;**  **}** |