**CPP Problem Design Example**

|  |
| --- |
| **Contributor︰ Pin-Shao Chen** |
| **Subject：Student Course System** |
| **Main testing concept：**   |  |  | | --- | --- | | **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：**  Design a course registration system, which can do the following things:  1. Add or delete a student.  2. Add or delete a course.  3. List the schedule of every student.  4. Error Handling  To be specific, please implement the commands in the Input/Output (each command will be separated by one space “ ”):  **Input/Output：**  Input some commands to operate course registration system. Each parameter separates by one space ” “. The following lists the legal commands and their format.   1. **AddStudent <Student\_name>:** Add a student. If the student already exists, output “**The student's name is duplicate.**” 2. **AddCourse <Student\_name> <Course\_name> <class\_time1> <class\_time2>… :** Add a course for the student, if the course will conflict with the schedule, output “**Course conflict.**” 3. **DelStudent <Student\_name>:** Delete a student. 4. **DelCourse <Student\_name> <Course\_name>:** Delete the course from the student's schedule. 5. **Print StudentList:** List all student's names in insertion order. Separate each student’s name by one space “ ”. Output “**The** **Students list is empty.**” if the StudentList is empty. 6. **Print <Student\_name>:** Print the schedule for the student in insertion order. Output **empty schedule** if the schedule is empty.   Example of **empty schedule** (for each line output the day with colon):  M:  T:  W:  R:  F:  Example of schedule:  M: 2:Math 3:Math 4:Math  T:  W: 1:DeepLearning 2:DeepLearning  R: 4:DeepLearning  F:  \*\* The pseudo code of output format.  cout << “M:”;  foreach course in Monday  {  cout << “ ” << time << “:” << course;  }  cout << endl;   1. **Print <Student\_name> <Course\_name>**   Print information of the course. Output in the order from M to F.  The output format: <**Course\_name**> **<class\_time1> <class\_time2>…**  Example:  >Print Jason Math  Math M2 M3 M4  **Schedule format:**  You can input the day of class\_time from M to F, the time of class\_time from 1 to 10.  (i.e., ‘R8 R9 R10’)   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **M** | **T** | **W** | **R** | **F** | | **1** |  |  |  |  |  | | **2** |  |  |  |  |  | | **3** |  |  |  |  |  | | **4** |  |  |  |  |  | | **5** |  |  |  |  |  | | **6** |  |  |  |  |  | | **7** |  |  |  |  |  | | **8** |  |  |  |  |  | | **9** |  |  |  |  |  | | **10** |  |  |  |  |  |   If the input tries to deal with a non-existent student, output “**The student's name does not exist.”**  If the input tries to deal with a course that is not on the schedule, output “**The course does not exist.”**  **<Course\_name> and <Student\_name> will be one word and only contain alphabet.**  Please output “Illegal command.” if the user inputs incomplete command, wrong command or too many parameters. Additionally, if the input parameter is not given a legal value such as “M11”, you should output “Illegal parameters.”  **Check the illegal command first, then parameters.**  **Sample Input / Output :**   |  |  | | --- | --- | | **Sample Input** | **Sample Output** | | AddStudent Jason  Print Jason  AddCourse Jason Math M2 M3 M4  AddCourse Jason DeepLearning W1 W2 R4  Print StudentList  AddStudent Mike1  Print Jason Math  Print Jason  DelStudent Jason  DelCourse Jason Math  Print StudentList  AddCourse Jason Science | M:  T:  W:  R:  F:  Jason  Illegal parameters.  Math M2 M3 M4  M: 2:Math 3:Math 4:Math  T:  W: 1:DeepLearning 2:DeepLearning  R: 4:DeepLearning  F:  The student's name does not exist.  The Students list is empty.  Illegal command. | | AddStudent Mike  AddStudent Mike  Print Mike  Print StudentList  DelCourse Mike Math  AddCourse Mike DeepLearning W1 W2 R4  AddCourse Mike Math W1 W2 W3  AddCourse Jason DeepLearning W1 W2 R4  Print Mike English  Print  Print Mike  DelStudent Jason  DelCourse Jason Math  Print Mike DeepLearning M11 | The student's name is duplicate.  M:  T:  W:  R:  F:  Mike  The course does not exist.  Course conflict.  The student's name does not exist.  The course does not exist.  Illegal command.  M:  T:  W: 1:DeepLearning 2:DeepLearning  R: 4:DeepLearning  F:  The student's name does not exist.  The student's name does not exist.  Illegal command. | |  |  | |
| **□ 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。 |
| **Other notes：** |