**CPP Problem Design Example**

|  |
| --- |
| **Subject: Observation Diary** |
| **Contributor: 陳宥潤,鄭永泰,范茗翔** |
| **Main testing concept: Class Design and Operator Overloading.**   |  |  | | --- | --- | | **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:**  You are making observations on several unknown creatures and you need to record the number of various body parts of them (e.g. How many legs, tails...) and generate diary observation logs.  You feel that manual diary logging is too cumbersome and want to design a program to automatically generate logs. You plan to design two classes **Diary** and **Creature** to handle this.   * Class **Diary** needs to implement the following membership functions. * **static void NewDay(string day):**   Set the current date code (A string of words. May not all necessarily be numbers.)   * Class **Creature** has two constructors. * **Creature(string name):**   Contains a creature named *name*.   * **Creature(string name, Creature(string name, Creature base):**   A creature that contains the same information as the *base*of each body part.  Suppose there was a Creature **c** named *sample* that has set the current date to "00". (Diary::NewDay("00"))   * Please achieve the following requirements by overloading the operators of class Creature. * **Obtain** the specified body part of Creature. (Overload operator [])   For example: c["leg"]: return to the body part of Creature c named "leg".   * **Assign** a value to a body part in Creature. (Overload operator =)   For example: c["leg"] = 3: set the number of body parts in Creature c named "leg" to 3 and record "sample's leg appeared (0 -> 3)" in the log.  \*\* Record "**<*name*>'s <*body\_part*> appeared (<*original\_value*> -> <*changed\_value*>)**" in the log each time this operation is called (the default value for the number of body parts is 0).   * **Increase** the number of specified body parts in Creature. (Overload operator +=)   For example: c["leg"] += 2: the number of body parts named "leg" increases by 2, so the number of body parts named "leg" becomes 5. And please record "sample's leg increased (3 -> 5)" in the log.  \*\* Record "**<*name*>'s <*body\_part*> increased (<*original\_value*> -> <*changed\_value*>)**" in the log each time this operation is called.   * **Reduce** the number of specified body parts in Creature (Overload operator -=)   For example: c["leg"] -= 2: the number of body parts named "leg" decreases by 2, so the number of body parts named "leg" becomes 3. And please record "sample's leg decreased (5 -> 3)" in the log.  \*\* Record "**<*name*>'s <*body\_part*> decreased** **(<*original\_value*> -> <*changed\_value*>)**" in the log each time this operation is called.  Note that the order of data will be legal and will not have negative or uninitialized number of body parts calculation.   * Class Creature also needs to implement the following member functions. * **void PrintStatus():** Prints the value of each body part of the organism.   For example: c.PrintStatus() should output:  sample's status:  leg \* 3   * **void PrintLog():** Prints the log information of the creature on a Diary basis since it was collected.   For example: c.PrintLog() should output:  sample's log:  Day 00  sample's leg appeared (0 -> 3).  sample's leg increased (3 -> 5).  sample's leg decreased (5 -> 3).  \*\* Output an empty line after PrintStatus() and PrintLog().  **Input:**  No inputs.  \*\* The main() function in your submission will be replaced when judging.  \*\* You can use the main() function in “Other Notes” to test your program.  **Output:**  The result of executing your program with the given main function.  **Sample Input / Output：**   |  |  | | --- | --- | | Main | Sample Output | | No inputs. | UA's log:  Day 0000  UA's leg appeared (0 -> 16).  Day 0102  UA's status:  UB's log:  Day 0102  UB's leg increased (16 -> 42).  Day 0227  UA's log:  Day 0000  UA's leg appeared (0 -> 16).  Day 0102  Day 0227  UA's leg disappeared (16 -> 0).  Day 0353  UA's leg appeared (0 -> 6).  UA's wing appeared (0 -> 4). | |
| **□ 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 more complex data types.** |
| **Expected solving time:**  50 minutes |
| **Other notes:**  int main() {  Diary::NewDay("0000");  Creature unknownA("UA");  unknownA["leg"] = 16;  Diary::NewDay("0102");  Creature unknownB("UB", unknownA);  unknownB["leg"] += 26;  unknownA.PrintLog();  Diary::NewDay("0227");  unknownA["leg"] = 0;  unknownA.PrintStatus();  unknownB.PrintLog();  Diary::NewDay("0353");  unknownA["leg"] += 6;  unknownA["wing"] += 4;  unknownA.PrintLog();  } |