**10902 CPP Final Exam**

|  |
| --- |
| **Contributor︰Wen, Yong-Wei** |
| **Subject：Design a Train** |
| **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：**  In 2080, people attempt to stop global warming via climate engineering catastrophically backfires. The failure of the plan causes the world to enter the ice age. The remnants of humanity have taken to a circumnavigational train, the Snowpiercer, run by transportation magnate Wilford.  Mr. Wilford arranges different uses for each carriage to ensure the maintenance of the daily. The train is connected by the carriage. For each carriage, he records the carriage name, the number of passenger and the name of passengers.  You need to implement two classes to help Mr. Wilford to arrange his train.   |  |  | | --- | --- | | Class: Train | Class: Carriage | | * Carriage\*: root * function: pushCarriage(string name, int passengerNum, string \*passengerName) * function: checkPassenger(string name) * function: deleteCarriage(string name) * function: getCarriage(std::string name) * default constructor() | public   * function: getPassengerNum() * function: getName() * function: printPassenger() * constructor(string name, int passengerNum, string\* passengerName) * destructor()   private:   * string: name * int: passengerNum * string\*: passengerName |   Carriage Function Implement:   1. int getPassengerNum(): return the number of the passenger in current Carriage. 2. string getName(): return the name of the current Carriage. 3. void printPassenger(): print the name of each passenger separated by one space (“ ”).   (i.e. “Melanie Wardell Kevin”)   1. constructor(): create the data of Carriage. 2. destructor(): delete the member variable which you have allocated the memory and print "Delete Carriage!"   Train Function Implement:   1. void pushCarriage(): Give the information of the Carriage. Use this to build a Carriage and push it to the end of the Train. 2. void checkPassenger(): Give the name of the Carriage. Find the Carriage with the same name and call its printPassenger. If it does not exist, print “Carriage does not exist!” 3. void deleteCarriage(): Give the name of the Carriage. Delete the Carriage with the same name and re-connect the train. If it does not exist, print “Carriage does not exist!” 4. Carriage\* getCarriage(): Give the name of the Carriage. Return the pointer of Carriage with the same name. If it does not exist, print “Carriage does not exist!”   **Input：**  You don’t need to deal with input format. The given file main.cpp will deal with all input format and catching (see void CreateTrain() in Notes). You can use the given main.cpp and enter the Sample Input to test your program.  The introduction of the Input:   1. First, input a positive number n which represent the number of Carriages. 2. For each Carriage, the first line input the carriage name and the number of passenger.  The second line input each passenger’s name.   **Notice again. The given main.cpp will deal with the catching of input. This is just a introduction of the input. You only need to implement the Train class and the Carriage class. You can directly using Sample Input to test your program.**  The input rule.   1. The input will not give you the same name of the carriage. 2. The length of the Train will not less than 3 carriages.   Additionally, there are a few points about our Judge testing.   1. The main.cpp in your submission will be replaced when judging. 2. You can use the main.cpp in “Other Notes” to test your program.   **Output：**   * When calling the function of checkPassenger(), you have to output the name of each passengers separated by space (“ ”). If the Carriage does not exist, you have to output “Carriage does not exist!” * When calling the function of deleteCarriage(). If the Carriage does not exist, you have to output “Carriage does not exist!” * When calling the function of getCarriage(). If the Carriage does not exist, you have to output “Carriage does not exist!”   **Sample Input / Output :**   |  |  | | --- | --- | | **Sample Input** | **Sample Output** | | 6  Engine 2  Melanie Bennett  Hospitality 3  Melanie Wardell Kevin  First\_class 2  Robert Lilah  Head\_bar 1  Ferami  Second\_class 3  Till Jinju Miles  Agricultural 2  Otto Sheila | Carriage name: Engine Passenger num: 2  Carriage name: Hospitality Passenger num: 3  Carriage name: First\_class Passenger num: 2  Carriage name: Head\_bar Passenger num: 1  Carriage name: Second\_class Passenger num: 3  Carriage name: Agricultural Passenger num: 2  The passenger in Hospitality:  Melanie Wardell Kevin  Carriage does not exist!  Delete Carriage!  Carriage name: Engine Passenger num: 2  Carriage name: Hospitality Passenger num: 3  Carriage name: First\_class Passenger num: 2  Carriage name: Head\_bar Passenger num: 1  Carriage name: Agricultural Passenger num: 2 | |
| **□** **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:**  #include "Train.h"  void printTrain(const Train& train)  {  Carriage\* cur = train.root;  while (cur != nullptr)  {  std::cout << "Carriage name: " << cur->getName() << "\tPassenger num: " << cur->getPassengerNum() << std::endl;  cur = cur->next;  }  }  void CreateTrain(Train& train)  {  int num;  std::cin >> num;  for (int i = 0; i < num; i++)  {  std::string name;  int passengerNum;  std::cin >> name >> passengerNum;  std::string\* passengerName = new std::string[passengerNum];  for (int j = 0; j < passengerNum; j++)  {  std::cin >> passengerName [j];  }  train.pushCarriage (name, passengerNum, passengerName);  }  }  int main()  {  Train train;  CreateTrain(train);  printTrain(train);  std::string checkName = "Hospitality";  std::cout << "The passenger in Hospitality:" << std::endl;  train.checkPassenger (checkName);  std::string deleteName = "Third\_class";  train.deleteCarriage (deleteName);  deleteName = "Second\_class";  train.deleteCarriage(deleteName);  printTrain(train);  return 0;  } |