**CSD201 PE INSTRUCTIONS**

**Follow the steps below to complete PE:**

1. Create a folder to save given projects, e.g. CSD\_given (1). Download the given materials to (1).
2. Complete the requirements of the test.
3. **Before submission:**
   1. **Clean and Build Project** (Shift+F11),
   2. Then rename the folder **dist** to **run**. (If the folder run already exists, delete it before renaming).
4. **Submission:** 
   1. To submit the project Q1, at first you must select the Question No: 1
   2. Browse and select the project folder (e.g. Q1, or Q1A, or Q2, …)
   3. Then click the **Submit** button.
5. **Do not use accented Vietnamese** when writing comments in programs.
6. **Do not add** new **import** statement(s) to given files.
7. Software tools must be used: **NetBeans IDE 8.x** and **Java JDK 1.8**.

**If at least one of the above requirements is not followed, the exam will get ZERO.**

* ***Notes:*** 
  + ***The input and expected output below are only used to test your codes.***
  + ***The input and expected output in the real testcases (for marking) are different with in the examples below.***
  + ***Do not hardcode with the given expected results.***

**Troubleshooting:**

If the given project (e.g. Q1) runs with error, you need to run "**Clean and Build Project**" (Shift+F11). If still error, try to rename or copy the project to other one, e.g. from Q1 to Q1X or Q1Y.

**Questions: (10 marks)**  
The given files already contain statements to implement a program for managing a **House** moving process. The structure of the main classes is as follows:

* **Class Item:** contains information about a furniture item, including:
  + name (e.g., "Table", "Chair", "Sofa")
  + weight (must be > 0, representing the weight of the item in kg)
  + length (must be > 0, indicating the number of meters the item occupies)
  + fragile is 1 (if the Item is **fragile**), or 0 (if this is **not fragile**)
* **Class Node:** includes info (a Item object) and next (a pointer for linking).
* **Class ItemList:** is a **singly linked list** that manages regular Nodes with complete information about the furniture in the house.
* **Class Truck**: is a **stack** structure (implemented as a linked list) where the info of the Nodes contains the furniture items loaded onto the truck. Stack has FIFO properties, the **stack** in this test is built using a singly linked list, the item is **push**ed to the **head**.
* **Class House:** is the main class of the program, containing a ItemList used to manage the furniture in a ItemList and a Truck corresponding to the truck transporting furniture.

Students are required to carefully read the provided code segments to fully understand the relationships between the classes and the functions within each class. The specific task of the test is to execute the following requirements:

**a. f1(): 2.5 marks – Load data**

To complete the requirement f1(), students need to:

1. Implement the function **addLast**(String name, int weight, int length, int fragile) in the **ItemList** class - list
2. Implement the function **push**(String name, int weight, int length, int fragile) in the **Truck** class - truck

The expected output used to test your code are as follows:

*ItemList Inventory: (Stove, 10, 2, 0) (Lamp, 5, 1, 1) (Table, 80, 3, 0) (Sofa, 50, 2, 0) (TV, 20, 1, 1) (Chair, 7, 1, 0)*

*Truck Contents: (Pot, 10, 1, 1) (Bed, 100, 2, 0)*

**b. f2(): 2.5 marks – Load the first fragile item onto the truck**

To complete the requirement f2(), students need to:

1. Implement the **deleteFirstFragile**() function to remove the first fragile item from the **list**.
2. Then **push** this item into the **truck**.

The expected output used to test your code are as follows:

*ItemList Inventory: (Stove, 10, 2, 0)* ***(Lamp, 5, 1, 1)*** *(Table, 80, 3, 0) (Sofa, 50, 2, 0) (TV, 20, 1, 1) (Chair, 7, 1, 0)*

*Truck Contents: (Pot, 10, 1, 1) (Bed, 100, 2, 0)*

*ItemList Inventory: (Stove, 10, 2, 0) (Table, 80, 3, 0) (Sofa, 50, 2, 0) (TV, 20, 1, 1) (Chair, 7, 1, 0)*

*Truck Contents:* ***(Lamp, 5, 1, 1)*** *(Pot, 10, 1, 1) (Bed, 100, 2, 0)*

**c. f3(): 2.5 marks – Load all fragile items onto the truck**

To complete the requirement f3(), students need to:

1. Remove all **fragile items** from the list.
2. Then push each removed item into the truck in the same order as they removed in the original list.

The expected output used to test your code are as follows:

*ItemList Inventory: (Stove, 10, 2, 0)* ***(Lamp, 5, 1, 1)*** *(Table, 80, 3, 0) (Sofa, 50, 2, 0)* ***(TV, 20, 1, 1)*** *(Chair, 7, 1, 0)*

*Truck Contents: (Pot, 10, 1, 1) (Bed, 100, 2, 0)*

*ItemList Inventory: (Stove, 10, 2, 0) (Table, 80, 3, 0) (Sofa, 50, 2, 0) (Chair, 7, 1, 0)*

*Truck Contents:* ***(TV, 20, 1, 1) (Lamp, 5, 1, 1)*** *(Pot, 10, 1, 1) (Bed, 100, 2, 0)*

**d. f4(): 2.5 marks – Compute total weight of items on the truck**

To complete the requirement f4(), students need to:

1. Remove all **fragile items** from the list. (as f3)
2. Push each removed item into the truck in the same order as they appeared in the original list. (as f3)
3. Compute and return the total **weight** of all items in the truck.

The expected output used to test your code are as follows:

*ItemList Inventory: (Stove, 10, 2, 0) (Lamp, 5, 1, 1) (Table, 80, 3, 0) (Sofa, 50, 2, 0) (TV, 20, 1, 1) (Chair, 7, 1, 0)*

*Truck Contents: (Pot, 10, 1, 1) (Bed, 100, 2, 0)*

*ItemList Inventory: (Stove, 10, 2, 0) (Table, 80, 3, 0) (Sofa, 50, 2, 0) (Chair, 7, 1, 0)*

*Truck Contents: (TV, 20, 1, 1) (Lamp, 5, 1, 1) (Pot, 10, 1, 1) (Bed, 100, 2, 0)*

*Total weight:* ***135***

===End of test===