Data Structures and Algorithms

Assignment Four

Due Friday 10 June 2016

Please ensure that relevant files are submitted in the correct assignment folder on AUT online by due date.

Questions:

1. Prepare a class called `Person` that represents information about a person such as their name, phone number and age. Its `hashCode` method should use the letters (char value) of the person's name added together. Then create a class called `HashTableWithChaining` for storing some E type objects, it should use chaining to resolve collisions and a load factor of 75% in which case it needs to expand capacity of the hash table by a factor of 2. The hash table should have methods to `add`, `remove` and `contains` an element, `size` and a suitable `toString` which prints out the entire table showing the chained links or nulls for entries where there is no elements. Test the class on `Person` objects and include a suitable `main` method which tests all features of the class including cases where it needs to expand capacity. **(20 marks)**

2. The next part of this assignment (Not related to Q1) is to create a program that uses the `Shipping` and `Ports` tables of the `testRestricted` database on the raptor2.aut.ac.nz server. These tables hold a list of ports and boat trips between ports. Obtain the class called `BoatTrip` from AUTOnline which represents a single boat trip between two different ports. The class keeps track of the departure and arrival times and names of the ports that the boat is travelling between, the cost of the trip and the boatID.

   Using the following UML diagram below a class called `Journey` that represents a set of boat trips between multiple ports. The class holds a `boatTripList` of `BoatTrip` objects. There are two constructors, one default and another that adds a list of boat trips to the `boatTripList`. The `addTrip` method should add a `BoatTrip` only if the Journey so far does not contain already contain its arrival port. The `containsPort` method should return true if the given port is in this Journey so far. The get methods return the start and end ports and dates or null if the Journey does not contain a boat trip. The `createClone` method returns a new Journey object instance and passes the `boatTripList` of current `BoatTrip` objects

by calling the second constructor. The `getTotalJourneyCost` should return the total cost of all the boat trips. The `toString` method should print out a string representation of all boat trips in this journey and the total cost.

Journey
-boatTripList : LinkedList<BoatTrip>
+Journey() +Journey(trips : List<BoatTrip>) +addTrip(trip : BoatTrip) : boolean +removeLastTrip() : boolean +containsPort(port : String) : boolean +getStartPort() : String +getEndPort() : String +getEndDate() : String +createClone() : Journey +getTotalJourneyCost() : int +toString() : String

(10 marks)

3. Prepare a class called `ShippingCenter` that reads from the `Shipping` and `Ports` tables in the database. The class does a recursive search of all the possible routes from one port to another. The constructor should connect to the database. The `readAllPorts` method should return a list of all the ports in the database from the `Ports` table. The `getAllJourneys()` method starts a journey from from the startPort to the endPort from a certain date. It calls the recursive method `findPaths()` that performs a depth-first search of all the possible paths that a Journey can take. Once complete a list of all the possible journeys is returned back to the user. The `findPaths()` method should query the database table `Shipping` to get all the paths that leave from the current port after the date that the trip arrives rather than reading in the entire database.

ShippingCompany
-conn : Connection -stmt : Statement -journeyList : List<Journey>
+ShippingCompany(dbUser : String, dbPassword : String) +readAllPorts() : List<String> +getAllJourneys(startPort:String,startDate:String, endPort:String) : List<Journey> -findPaths(currentJourney:Journey, startDate:String, startPort:String, endPort:String) :void +close() : void +main(args : String[]) : void

(20 marks)