Data Structures
Fall 2016, Final Project

The target of the final project is to finish the implementation of a new complete system for the U-bike company.

U-bikes and Rental stores

There are 4 types of bikes: Electric, Lady, Road, and Hybrid. The bikes are abstracted as class UBike, and are dynamically (de)allocated in the system. The pointers to the bikes are stored in a hash table and customized min heap structures. On the other hands, the rental stores are located in 12 MRT stations including Danshui, Hongshulin, Beitou, Shilin, Zhongshan, Xinpu, Ximen, Liuzhangli, Muzha, Guting, Gongguan, and Jingmei stations. In each station, there are 4 min heaps for the four types of bikes and a min heap for rented U-bikes. These heaps are based on the bike mileages, and we use a variable heapIndex to record the position of the bike in its corresponding heap.

Rental charges

The system constructs a graph where each node represents a station, and each edge represents a road that connects 2 neighboring stations. The weight of each edge is defined as the road's discount distance which is determined by the company and is provided in an input file(map.in) of the following format.

Format	"Station A" "Station B" "Discount distance of the road AB"
Example	Danshui Hongshulin 49 Danshui Beitou 58 Hongshulin Beitou 1 …

The company offers a discount to those who ride within(≤) the shortest discount distance between the stations where you rent and return the bike. Rent rates are listed in the following table.

Rate Table	Electric	Lady	Road	Hybrid
Discount (≤)	\$25/mile	\$20/mile	\$10/mile	\$15/mile
Original (>)	\$40/mile	\$30/mile	\$20/mile	\$25/mile

For example: **(1)** You return Lady-bike to MRT station, and 15 miles you drove was within the shortest threshold distance. The charge is : 15 * 20; **(2)** You return Road-bike to MRT station, and 10 miles you drove exceeds the shortest threshold distance. The charge is : 10 * 20;

License and hash table

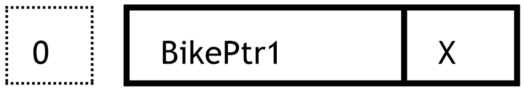
In order to locate bikes quickly by license (5 alphanumeric characters A..Z and 0..9), a hash table is used. If a bike is owned by the company, the hash table will contain the pointer to the bike. The hashing function is defined as follows

1. '0'~'9' correspond to 0-9 and 'A'~'Z' correspond to 10-35.
2. Let 5-character license be denoted as x i.e., **x[0] x[1]…x[4]**.
3. **S(0) = x[0]; S(n) = S(n-1) * 29 + x[n]**
4. 6th to 13th bits (8-bit) of S(4) is used as the address in the hash table.

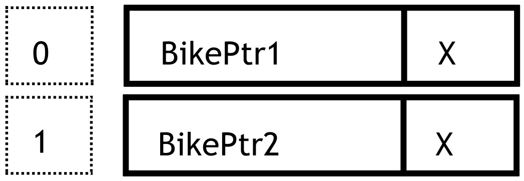
00A03	**x[0]**	**x[1]**	**x[2]**	**x[3]**	**x[4]**
License Tag	**0**	**0**	**A**	**0**	**3**
S(4)	8413 = 001 **00000110** 11101				
Address	**00000110** = 6				

5. Collision is handled by chaining(singly linked list), and the new bike is inserted to the **front**.
   For example:

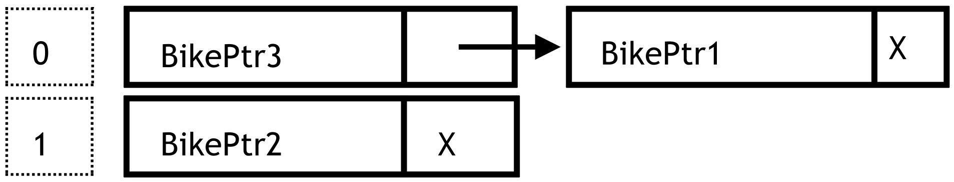
If hash(bike1's license) = 0, we push BikePtr1 to the hash table index 0.



If hash(bike2's license) = 1, we push BikePtr2 to the hash table index 1.



If hash(bike3's license) = 0, collision occurs and we solve it by chaining.



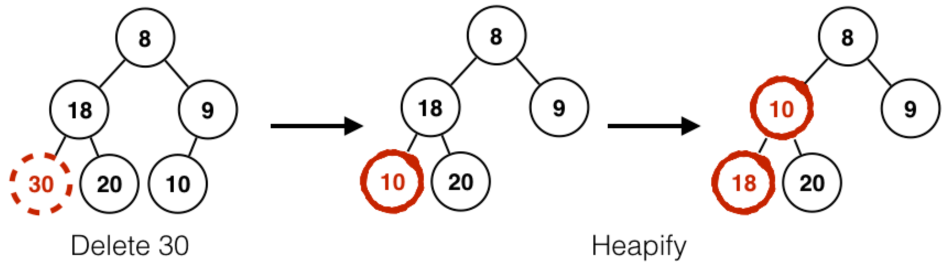
Min heap that can delete any specific node

The delete operation (removeUBikePtr) used in the heap is customized so that any node can be removed from the heap. For each deletion, you need to replace the node to be deleted with the last node and perform following two operations.

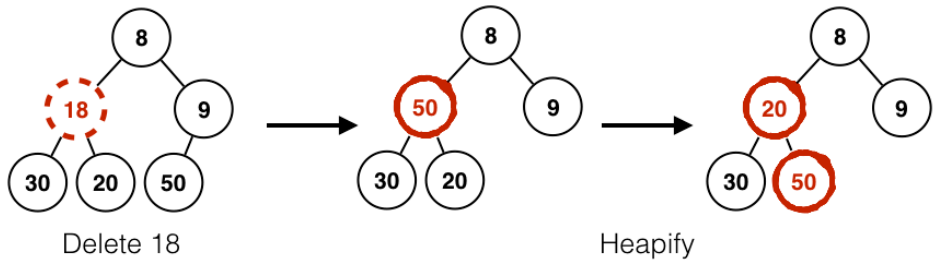
(1) **bubble-up**: If the bike pointed by the current node has lower mileage than its parent, the two nodes are swapped (Example 1.).

(2) **bubble-down**: If the bikes pointed by L-child or R-child have lower mileage, we swap the current node and the one with lower mileage (Example 2.). However, when the mileages of L-child and R-child are equal, we swap the current node and the L-child. (Example 3).

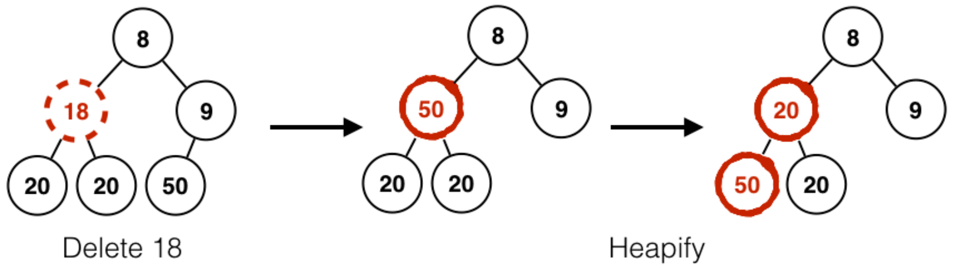
Example 1 : **bubble-up**, parent(18) is larger than 10 , swap 18 and 10.



Example 2 : **bubble-down**, L-child(30) and R-child(20) are smaller than 50, swap 50 and 20 (the smaller one).



Example 3 : **bubble-down**, if L-child(20) and R-child(20) have equal value and are smaller than 50, swap L-child and 50.



Input Commands

The input file cm.in consists of the following 5 kinds of commands, each of which implies calling the corresponding function of UBikeSystem.

1	**NewBike classType license mile station** Call UBikeSystemADT::NewBike(classType, license, mile, station);
2	**JunkIt license** Call UBikeSystemADT::JunkIt(license);
3	**Rent station classType** Call UBikeSystemADT::Rent(classType, station);
4	**Returns station license returnMile (total current mileage)** Call UBikeSystemADT::Return(station, license, returnMile);
5	**Trans station license** Call UBikeSystemADT::Trans(station, license);

Submission

The **team leader** is asked to submit to ilms an archive, named "**ds2016f_final.zip**", of all your source code files that can be compiled properly using readonly/makefile. You are allowed to submit your archive **TWICE**. The 2 deadlines are **2016/12/29** and **2017/01/05**. If you get 100 in the first submission, there is no need to submit your code again.

Evaluation

For each test case, the evaluation is done by comparing the final status of UBikeSystem with the correct answer (ans.out). Please refer to main.cpp for more details.

- [60%] 3 public test cases, 20% each.
- [40%] 4 **hidden** test cases, 10% each.
- Note1: Do **initialize** allocated memory space that you may use.
- Note2: Do test your program **thoroughly** using self-designed test data to avoid runtime error.

Detailed C++ Sub-routines

UBikeSystem
void **InitDistTable** (std::string MapFile) Parse the discount distance file MapFile, and calculate pairwise shortest distances of the 12 stations. The distances will be used to calculate the rental charges.
void **NewBike** (std::string classType, std::string license, int mileage, std::string station); The parameters license, mileage and classType represent the unique license number, mileage and class respectively for a new bike purchased by the U-bike company. The function creates a new bike accordingly and inserts the pointer into appropriate positions in the hash table and classType heap in station.
void **JunkIt** (std::string license); The function deletes the node of the bike license, and its pointers in the hash table and the heap. Specifically the function find the bike pointer by looking up the hash table, and remove the pointer from the hash table and from the containing heap. If the bike does not belong to the company(not in the hash table) or is being rented, however, the operation is ignored.
void **Rent** (std::string classType, std::string station); This procedure performs all the 3 steps involved in renting a bike of type classType from station. (1) Find the pointer(ubptr) of the bike with the min milage in classType heap in station. (2) Delete the bike pointer from the classType heap in station, and add it to the "Rented" heap in station. (3) set ubptr->isRented to true. However, if there is no free bike available, nothing should be done.
void **Return** (std::string station, std::string license, int returnMile); This function returns the bike license to station.(let say ubptr) returnMile represents the mileage on the bike upon its return. (1) Calculate the rental which is accumulated to the net of the u-bike company. (2) Update the mileage of the bike license, and set its isRented false. (3) ubptr is deleted from the "Rented" heap in ubptr->station and is added to the corresponding heap ubptr->classType in ubptr->station. Please note that the returned bikes must be rented.
void **Trans** (std::string station, std::string license) This procedure transfers the bike license to station. The procedure contains the following 2 steps: (1) Find the pointer of the bike license using the hash table. Let say the one pointed by ubptr. (2) Delete ubptr from the heap that contains it, and add ubptr to the corresponding heap in station. However, if the bike does not belong to the company or is being rented, the procedure is ignored. If the bike is transferred to the same station (station==ubptr->station), steps (1), (2) should be done.
void **ShutDown**() All bike pointers should be deleted before the program exits.

PriceTable
void **calcAllPairsShortestPath**(std::string mapFile); Parse the discount distance of each road from mapFile, and calculate all pairs shortest paths.
int **calcPrice**(int mileageDiff, std::string bClass, std::string from, std::string toStation); Calculate and return the rental charges: The U-Bike of type bClass is rented from from and returned to toStation. The charge can be calculated by the multiplication of the appropriate rate and mileageDiff.

UBikeHashTable
void **addUBikePtr**(UBike* ubptr); Add the pointer ubptr to the hash table.
UBike* **findUBikePtr**(std::string license, bool toRemove); Find the pointer to the Ubike license. If toRemove is true, remove the pointer from the hash table.

UBikeHeap
void **addUBikePtr**(UBike* ubptr); Add the pointer ubptr to the min heap.
UBike* **removeUBikePtr**(int heapIndex); Remove the pointer at heapIndex in the min heap.