Data Structures

Spring 2014, Final Project

This final project requires the development and implementation of a new complete system for automating transactions of the Taipei U-bike. All types used in this assignment are defined in the appendix.

The Taipei U-bike rental stores are located in several MRT stations. Each MRT station can be denoted by one of the following names.

Danshui(淡水)	Hongshulin(紅樹林)	Beitou(北投)	Shilin(士林)
Zhongshan(中山)	Xinpu(新埔)	Ximen(西門)	Liuzhangli(六張犁)
Muzha(木柵)	Guting(古亭)	Gongguan(公館)	Jingmei(景美)

A variable named 'StationType' is defined to be one of the MRT station. All the MRT stations form a graph, so the graph contains 12 nodes. The edge weight of the graph is defined to be the distance of the path between neighboring MRT stations. Each MRT station can be reached from any other MRT station. The information of paths is represented by the following format, and is stored in a provided file.

Format	"MRT Station A" "MRT Station B" "Distance From A to B"
e.g.	Danshui Hongshulin 49 Danshui Beitou 58 Hongshulin Beitou 1 ….

Bikes are represented by nodes (of type BikeType). All the nodes are stored in a hash table and referenced by other data structure.

Bike rental charges are classified under two ranges:

	Discount and Origin
Class Electric -	$30/mile and $40/mile
Class Lady -	$25/mile and $30/mile
Class Road -	$15/mile and $20/mile
Class Hybrid -	$20/mile and $25/mile

We offer a discount to those who drive within shortest path distance. The cost of mileages that used over shortest path distance is calculated with original charge.

Information about the operation of various rental MRT Stations is maintained in the following array of records:

OfficeType Table[12];

where entries 0..11 correspond to the cities. Note that this array can be indexed by using the StationType: Danshui, Hongshulin, Beitou, Shilin, Zhongshan, Xinpu, Ximen, Liuzhangli, Muzha, Guting, Gongguan and Jingmei.

In order to locate bikes quickly by only providing license tag (5 alphanumeric characters A..Z and 0..9), a Hashing Table is used. The Hashing Table will enable us to check whether a license tag corresponding to a bike is owned by Taipei U-bike or not. If it is, the Hashing Table will contain a pointer to a bike record corresponding to the license tag. For this purpose, the field 'Ptr' in a node of type 'HNodeType' in the Hashing Table is used which points to a node of type 'BikePtr'. Following this pointer will allow us to go from the Hashing Table to that particular node.

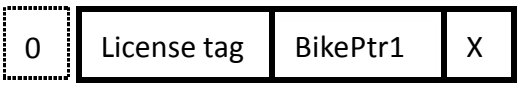
The Hashing function is defined as follows:

1. '0'~'9' correspond to 0-9 and
   'A'~'Z' correspond to 10-35.
2. Let 5-character license tag be donated as x i.e., x[0] x[1]…x[4].
3. $\begin{cases} S(0) = x[0]; \\ S(n) = S(n-1) * 31 + x[n]; \end{cases}$
4. 11th to 18th bits (8-bit) of S(4) is used as the address to hashing table.

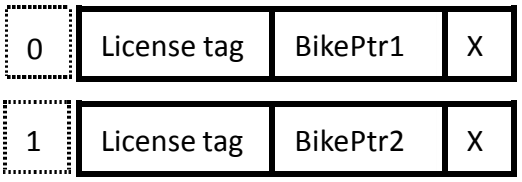
Moreover, overflow is handled by chaining. (00A03 entry is 9)

For example:

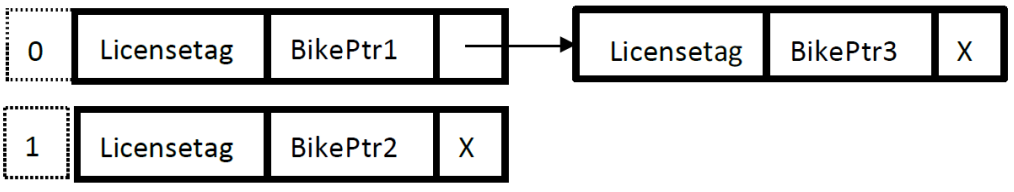
if hash(bike1's license tag) = 0, we push bike1 ptr to the hash table index 0



if hash(bike2's license tag) = 1,we push bike2 ptr to the hash table index 1



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



BikeNum is a variable of type ’integer’ that indicates the number of bikes owned by Taipei U-bike. Whenever a new bike is added to the Taipei U-bike’s collection, BikeNum is increased by one. Similarly, when a bike owned by the Taipei U-bike is scrapped, BikeNum is decreased by one and the appropriate modifications have to be made in the Hashing Table.

For each MRT Station, we will keep four heaps (HElectric, HLady, HRoad and HHybrid) for four types (Electric, Lady, Road and Hybrid) of bikes. Note that the entries in the heap are pointers of type BikePtr. The ordering in these heaps is with respect to the mileage of the bikes (largest value on the top of heap). In each node of type BikeType we use the variable Cursor to point to the position in the heap where there is a pointer to the node.

Implement the following C/C++ subroutines:

BikePtr NewBike (LicenseType License, int Mile, ClassType Class, StationType Station)

The parameters License, Mile and Class represent the license number, mileage and class respectively for a new bike purchased by the Taipei U-bike. In this and all succeeding subroutines, Station is of type StationType and indicates the designated MRT Station for the newly purchased bike. The function creates a new node for the bike and inserts it to the appropriate positions in the lists. The value returned by the function is a pointer to the newly created node.

BikePtr SearchBike (LicenseType License)

The parameter License represents the license number. Hash table is searched and a pointer to the bike with license number License is returned. If a null pointer is returned, the bike is not found.

int JunkBikePtr (BikePtr Bike)

The function performs the necessary alterations and data movements involved with the deletion of node that Bike points to. If the operation is successful, the function returns zero. Otherwise, it returns an appropriate integer value which indicates why it has not been possible to carry out the operation. Possible reason is that the bike does not exist.

The way to remove certain node in the heap should be carefully considered. The complexity of this operation should be O(logn), where n is the number of nodes in the heap.

void TransBikePtr (StationType Station, BikePtr Bike);

This procedure performs the necessary actions for transferring bike Bike to Station. To be eligible for transfer, Bike must be on free status. Node Bike is inserted in the heap for its class in Station. There may be occasion when the condition specified for a valid transfer is violated (e.g., Bike is on rental status). Error messages are printed and control is returned to the calling program

void RentBikePtr (StationType Station, BikePtr Bike);

This procedure performs all the necessary actions involved in renting the bike Bike from Station:

1. Change the status of Bike (i.e., free to rented).
2. Increment the appropriate one of NumTrsEletric, NumTrsLady, NumTrsRoad, NumTrsHybrid for Station.
3. Pointer Bike must be deleted from the corresponding heap of free bikes of Station and pointer Bike must be added to the corresponding heap of HRent of Station.

float Returns (StationType Station, BikePtr Bike, int ReturnMile)

1. This function performs all the necessary actions in returning the bike Bike to Station, ReturnMile represents the mileage on the bike upon its return.
2. Update Bike→Mileage by ReturnMile and change Bike→Status (i.e., rented to free).
3. Pointer Bike must be deleted from HRent in MRT Station Bike→Station and a pointer to Bike must be added to the corresponding heap of free bikes of Station.
4. If the returned Station, is not the same as the associated station, (Bike→Station), Trans procedure is required to be performed.
5. Calculate the rental charges from (Bike→Mileage, ReturnMile)

Add these charges to the Station and the Taipei U-bike. Remember, you have to update the Mileage of the Bike with ReturnMile.

void Inquire (LicenseType License);

This procedure prints out a summary report on the bike with license number License. The report includes all the information contained in the field of the node representing the specified bike. If the bike with this license is not found, an appropriate message is printed and control returned to the calling program.

void StationReport (StationType Station);

This procedure prints out a complete report on all the bikes associated with the Station. The initial output is subdivided into the following two categories: free bikes and rented. For each bike, the information displayed is the same as that printed by procedure INQUIRE. The report also includes the total number of bikes in free and rented status; and the value of Net, NumTrsEletric, NumTrsLady, NumTrsRoad and NumTrsHybrid for Station.

void UbikeReport();

This procedure prints out a complete report on all the Taipei U-bike's stations. For each MRT Station, the information display is that printed by procedure StationReport. There is a summary of the aggregate totals for the Taipei U-bike. The appropriate titles have to be included in the output to simplify reading.

void NetSearch (StationType Station);

This procedure prints out the Station Net. Net is the amount of income collected from renting the bikes. You have to print each BikeType Net, and display total Net of MRT Station.

void HashReport ();

Print the information about the Hash Table

MAIN program

Develop a main program to perform the various services for the bike rental Taipei U-bike. The various transactions are simulated by input data lines, and the map of MRT stations is given by a provided file. That is, your main function should be of the form.

```
int main(int argc, char * argv[])
{
	if(argc!=3)
	{
		cout << "arguments are incorrect" << endl;
		return 0;
	}
	//argv[1] is the transaction file
	//argv[2] is the map file

	//your program..

	return 0;
}
```

Steps to be taken for each ACTION are described as follows:

1. NewBike **Class License Mile StationName**
   (i) Create a node,
   (ii) Put the License in the Heap and Hash table,
   (iii) Print a message ”New bike is received by Station **StationName**.”

2. JunkIt **License**
   (i) Search the bike with license number License using hash table
   (ii) Delete the bike (perform function JunkBikePtr),
   (iii) Print error message if the bike does not exist,
   (iv) Print a message ”Bike **License** is deleted from **StationName**.”

3. Rent **StationName Class**
   (i) Find a free bike of bike type **Class** with the largest mileage in **StationName**
   (ii) If we cannot find such a bike, print an error message ”No free bike available.”
   (iii) Perform procedure Rent for renting a bike.
   (iv) Print a message “A bike is rented from **StationName**.”

4. Returns **StationName License Mile** (total current mileage)
   (i) Perform function Returns
   (ii) Print a message “Rental charge for this bike is $$$.”

5. Trans **StationName License**
   (i) Move the bike with License to lot StationName
   (ii) Print a message ”Bike **License** is transferred to **StationName**.”

6. Inquire **License**
   Print the information about License

7 StationReport **StationName**
   Print the information about StationName

8. UbikeReport
   Print the information about the Taipei U-bike

9. NetSearch **StationName**
   Print the information about StationName Net and each BikeType Net.

10. HashReport
   Print the information about the Hash Table

Submission:

請同學們個別建立自己的資料夾，名稱請使用自己的學號，

並上傳到 ILMS。

Filename format:

學號_version.zip/rar

例如： 102062546_v1.zip/rar

Notice:

1. 程式要利用 main 參數吃外面的 input 檔案(檔案名稱由助教輸入)
2. Output 檔案必須名叫"output.txt"
3. 可執行檔和 source code 都要上傳
4. APPENDIX 為參考 Data Type，同學不一定要按照上面的實作，只要你的結果與助教的相同。

APPENDIX : DATA Type

```
enum StatusType {Free, Rented};
enum ClassType { Electric, Lady, Road, Hybrid};
enum StationType {
     Danshui, Hongshulin, Beitou,
     Shilin, Zhongshan, Xinpu,
     Ximen, Liuzhangli, Muzha,
     Guting, Gongguan, Jingmei };

typedef char LicenseType[5];

struct BikeType{
     LicenseType License;
     StatusType Status;
     int Mileage; /* most recently rented mileage */
     int Cursor; /* cursor to the entry in heap where there is a pointer to this node */
     StationType Station;
     ClassType Class;
};

typedef struct BikeType *BikePtr;

struct HeapType{
     BikePtr Elem[256]; /* use array to implement heap, and each node in the heap is a pointer*/
     int Number;
};
struct OfficeType{
     int Net; /* total income of  station */
     int NetEletric;
     int NetLady;
     int NetRoad;
     int NetHybrid;
     int NumTrsEletric; /* number of electric bikes */
     int NumTrsLady; /* number of lady bikes */
     int NumTrsRoad; /* number of road bikes */
     int NumTrsHybrid; /* number of hybird bikes */
     HeapType HRent;
     HeapType HElectric;
     HeapType HLady;
     HeapType HRoad;
     HeapType HHybrid;
};

struct HNodeType {
     LicenseType License ;
     BikePtr Ptr; /* point to the node representing the bike */
     struct HNodeType *next_ptr; /*point to the next node in the chaining list*/
};
struct HTableType {
     HNodeType table[256];
     /*since each entry in the hash table is a pointer, it will be initialized as NULL;*/
     int BikeNum; /* the number of bikes in the hash table */

};
```