Data Structures
Spring 2016, Final Project

This final project requires the development and implementation of a new complete system for the Taipei U-bike.

Rental Stores

The Taipei U-bike rental stores are located in several MRT stations. Each MRT station has 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 as one of the MRT station. The 12 stations form a graph where each node represents a station. The weight of each edge is defined as the distance between 2 neighboring stations, and is provided in an input file of the following format.

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**) that are stored in a binary search tree and referenced by other data structure. We offer a discount to those who drive **within shortest path distance** between the stations where you rent and return the bike. Bike rental charges are listed in the following table.

Discount and Original Price

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

For example:

(1) You return Lady-bike to MRT station, and 15 miles within the shortest path.
The charge is : 15 * 25;

(2) You return Road-bike to MRT station, and 10 miles without shortest path.
The charge is : 10 * 20;

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 binary search Tree.

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

License Tag

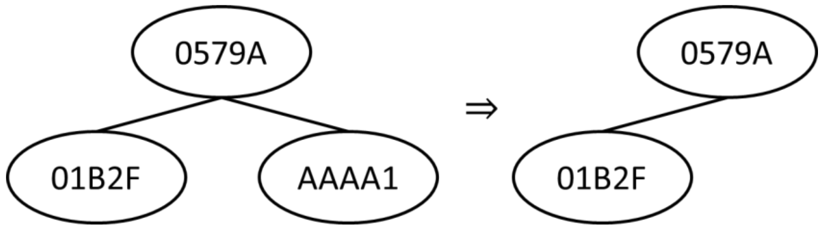
In order to locate bikes quickly by only providing license tag (5 alphanumeric characters A..Z and 0..9), a binary search tree is used. The binary search tree will enable us to check whether a license tag corresponding to a bike is owned by Taipei U-bike or not. If it is, the binary search tree will contain a pointer to a bike record corresponding to the license tag. For this purpose, the field ’Ptr’ in a node of type **’BNodeType’** in the binary search tree is used, which points to a node of type **’BikePtr’**. Following this pointer, we can access particular node from the binary search tree.

The rule of binary search tree is the following:

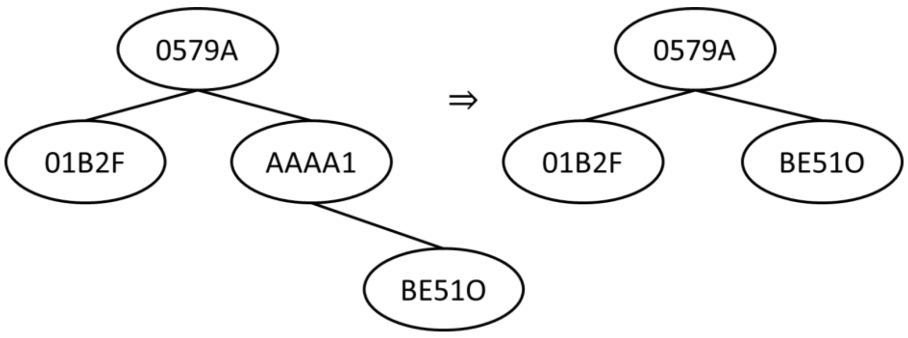
1) Use the ASCII Code to implement the “Lexicographical order”.
   e.g. 00000<00001<…<00009<0000A<…<0000Z
2) When you need to delete the nodes in the binary search tree, if the node B is a leaf in the tree, then just delete the node B.
   Otherwise, if the node B is non-leaf, then there are 2 case.
   1. If node A only has right child. Replace A with A->right child.
   2. If node A has left child, replace A with the left child which has the largest value.

   e.g. delete the node AAAA1 from the following tree

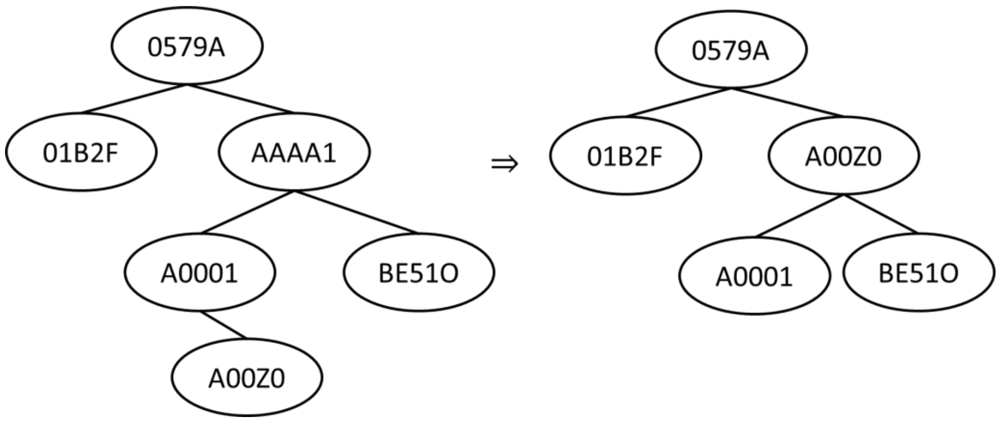
Leaf case:



Only right child case:



Replace with the left child which has the largest value case:



Implement the following C/C++ subroutines:

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

The parameters License, Mile and Class represent the unique license number, mileage and class respectively for a new bike purchased by the Taipei U-bike. The license of each bike is unique. The function creates a new node for the bike and inserts the node to appropriate positions in the binary search tree and the corresponding heap (HElectric, HLady, HRoad and HHybrid) of Station. The value returned by the function is a pointer to the newly created node.

e.g. NewBike(NTHU0, 10, Lady, Jingmei)

Then add a Lady bike with license NTHU0 to Jingmei Station.

BikePtr SearchBike (LicenseType License**)**

The parameter License represents the license number. Binary search tree 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.

e.g. SearchBike(NCTU0)

Then search a bike with license NCTU0 in the binary search tree.

int JunkBikePtr (BikePtr Bike)

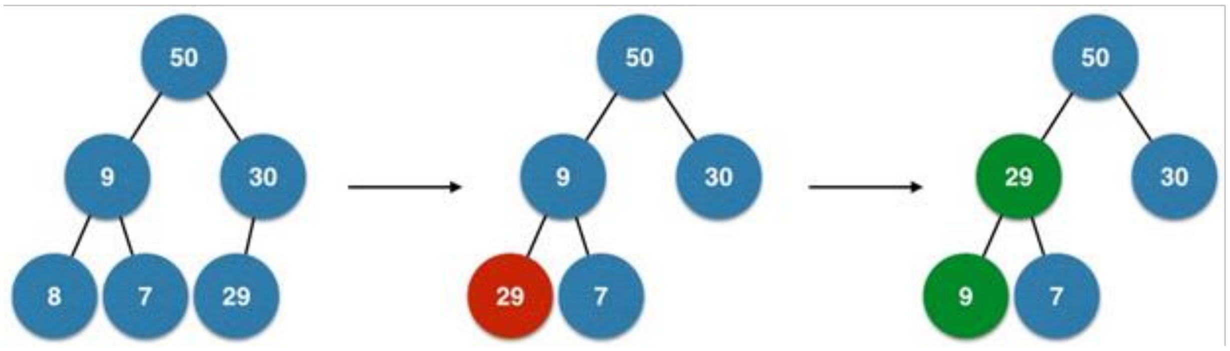
The function deletes the node that Bike points to. If the operation is successful, the function returns zero. Otherwise, it returns an appropriate integer value which indicates if the operation is successfully carried out. Possible reason is that the bike does not exist.

The way to remove certain node in the heap should be carefully considered. For each deletion, you need to replace the node to delete with the last node and perform following two operations.

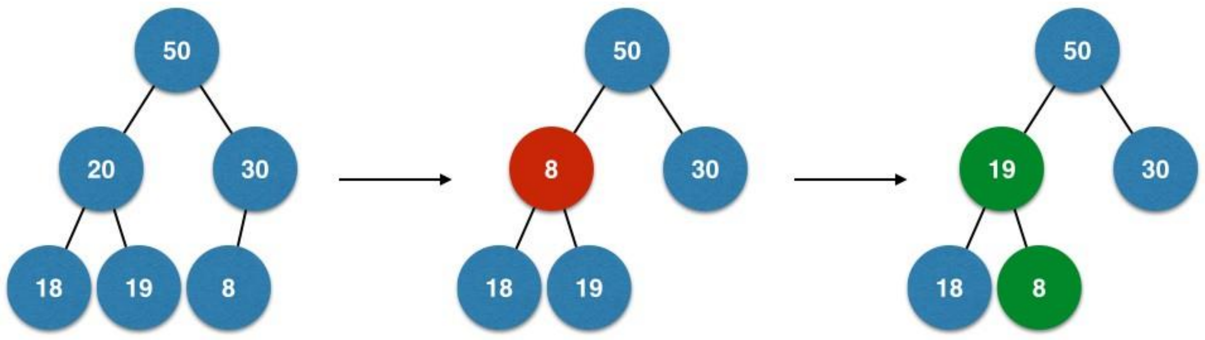
(1) **bubble-up**: check if the parent is smaller than current node or not, if yes, we exchange the parent with current node.

(2) **bubble-down:** check if the L-child and R-child are bigger than current node or not, if they have equal value and bigger than current node, we exchange L-child with current node.

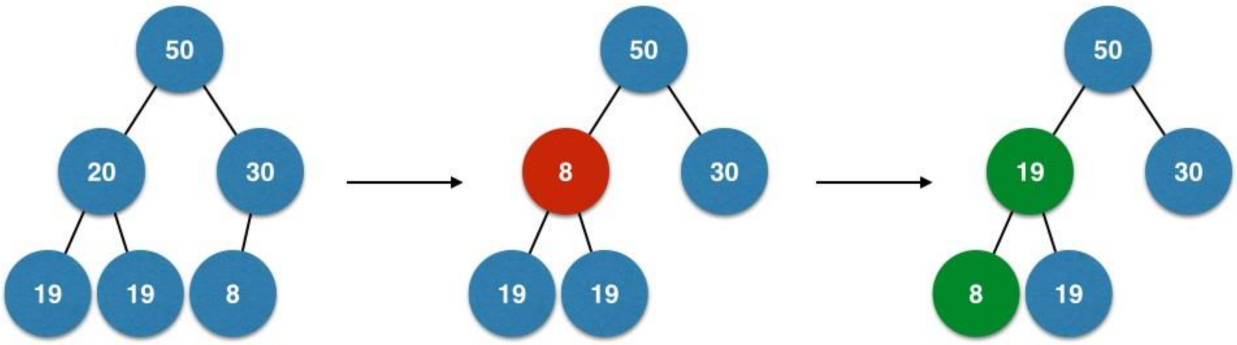
Example 1: **bubble-up**, parent(9) is smaller than 29, exchange 29 with 9



Example 2: **bubble-down**, L-child(18) and R-child(19) are bigger than 8, exchange 8 with 19(the bigger one).



Example 3: **bubble-down**, if L-child and R-child are bigger than 8, and have equal value, we exchange L-child with 8.



e.g. If the BikePtr point to NTHU0 and NTHU 0 belongs to Jingmei station now.

JunkBikePtr(BikePtr)

Then delete NTHU0 from the Jingmei station.

void TransBikePtr (StationType Station, BikePtr Bike)

This procedure transfers Bike to Station, and Bike will belong to Station. Bike must be on free status. Node Bike is inserted in the heap of 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.

e.g. If the BikePtr point to NTHU0 and NTHU 0 belongs to Jingmei station now.

TransBikePtr(Ximen,BikePtr)

Then transfer NTHU0 from Jingmei station to Ximen station.

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. Delete pointer Bike from the corresponding heap of free bikes in Station. Add Bike to the HRent in Station.

   e.g. If the BikePtr point to NTHU0 and NTHU 0 belongs to Jingmei station now.

   RentBikePtr(Jingmei,BikePtr)

   Then NTHU0 is rent now.

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

1. This function returns Bike to Station. ReturnMile represents the mileage on the bike upon its return.
2. Update Bike->Mileage by ReturnMile and change Bike->Status.

3. Bike must be deleted from HRent in Bike->Station and Bike must be added to the corresponding heap of free bikes in Bike->Station.
4. Add the charge to Bike->Station and the bike will be also **returned to Bike->Station, instead of Station.**
5. Calculate the rental charges from (Bike->Mileage, ReturnMile)

   e.g. If the BikePtr point to NTHU0 with class=Lady, mileage=10 and NTHU0 belongs to Jingmei station now.

   Returns(Ximen, BikePtr, 20)

   Then bike NTHU0 is returned to Ximen station and update NTHU0->mileage to 20. NTHU0 is removed from the maxheap HRent in Jingmei station and add NTHU0 to maxheap HLady in Jingmei station. If the shortest path from Jingmei to Ximen is 10, the rental fee is 10*25. **However, bike NTHU and the rental fee are still belongs to Jingmei station, and you can rent NTHU0 in Jingmei station again, not in Ximen station.**

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 License is not found, an appropriate message is printed and the control is returned to the calling program.

e.g. Inquire(NTHU0)

Then print the summary report of NTHU0.

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, NumElectric, NumLady, NumRoad and NumHybrid for Station.

e.g. StationReport(Jingmei)

Then print the station report of Jingmei station.

void UbikeReport()

This procedure prints out a complete report on all the Taipei U-bike's stations. For each MRT Station, the information displayed 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.

e.g. UbikeReport()

Then print the report of all the Taipei U-bike's stations.

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.

e.g. NetSearch (Jingmei)

Then print the Net of Jingmei station.

void BReport ()

Print the information about the Binary search tree by 2 lines. The first line is the preorder traversal and the second line is inorder traversal of the binary search tree.

e.g. BReport ()

Then print the preorder and the inorder of the binary search tree.

void Main ()

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. Do use main arguments to feed the input/output file names.

```
int main ( int argc, char * argv[] ){

    if (argc!= 4){
        cout << "arguments are incorrect" << endl;
        return 0;
    }
    //argv[1] is the name of the transaction file
    //argv[2] is the name of the map file
    //argv[3] is the name of the output 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 binary search tree. (iii) Print a message ”New bike is received by Station StationName.”
2.	JunkIt License (i) Search the bike with license number License using binary search tree. (ii) Delete the bike in the heap and binary search tree. (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 is available.” (iii) Perform function RentBikePtr 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 StationName. (ii) Print a message ”Bike License is transferred to StationName.”
6.	Inquire License (i) Print the information about License.
7.	StationReport StationName (i) Print the information about StationName.
8.	UbikeReport (i) Print the information about the Taipei U-bike.
9.	NetSearch StationName (i) Print the information about StationName Net and each BikeType Net.
10.	BReport (i) Print the preorder and inorder traversal of the binary search tree.

Submission

Submit to ilms an **archive** of all your source code files, executable and README files.

FILE	Name & Content
Archive	[Student_ID]_version.zip (e.g. 102062546_v1.zip or 102062546_v2.zip …) 1. Source Codes 2. Executable File 3. README
1. Source Codes [4%]	Put all your codes in a **single directory** called **src** where main function is written in a file named **main.cpp**. You can only use **.h/.cpp** files even if you write c code.
2. Executable File [3%]	**main.exe** or **main.out**
3. README [3%]	A txt file, **README.txt**, that contains the following The operation system and integrated development environment [IDE] where you develop the system: e.g. [Xcode in Mac OS X 10.9], [Visual Studio 2012 in Windows 7], [Eclipse in Ubuntu 12.04.2 LTS], [Makefile in Mac OS X 10.10]…

Evaluation

A. [10%] Submit your archive that satisfies the requirements.
B. [20%] Pass a public test case.
C. [20%] Pass a public test case that consists of **NewBike** and **Inquire** commands.
D. [20%] Pass a public test case that consists of **NewBike**, **Inquire**, **JunkIt** and **BReport** commands.
E. [30%] Pass several hidden comprehensive test cases.

p.s. If you do not have sufficient time to implement all functions, you are suggested to start from the 4 commands: **NewBike**, **Inquire**, **JunkIt** and **BReport**.

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 StationType{
        int Net; /* total income of station */
        int NetElectric;
        int NetLady;
        int NetRoad;
        int NetHybrid;
        int NumElectric; /* number of electric bikes */
        int NumLady; /* number of lady bikes */
        int NumRoad; /* number of road bikes */
        int NumHybrid; /* number of hybrid bikes */
        HeapType HRent;
        HeapType HElectric;
        HeapType HLady;
        HeapType HRoad;
        HeapType HHybrid;
};
```