University of Regina
Department of Computer Science
CS210-001 Data Structures and Abstractions (Winter 2016)

Assignment 3 (Assigned: Tuesday, February 16, 2016, Due: Tuesday, March 8, 2016)

Exercises

(**No** programming required, 2 + 2 = 4 Marks)

1. Using **pseudocode** or **C++** code, and the following ADT unsorted list operations, write a procedure

void CopyList(UnsortedType &L, UnsortedType &M)

to **copy** the elements of an unsorted list *L* to an empty unsorted list *M*, such that *L* is in its **original** state before the function ends. Thus, *L* may be modified during the function, but it must be restored to its **original** state before the function ends. Note that you may **not** access the internal data members of the unsorted list ADT.

void	*MakeEmpty()*	*void*	*InsertItem(ItemType item)*
bool	*IsFull() const*	*void*	*DeleteItem(ItemType item)*
int	*GetLength() const*	*void*	*ResetList()*
void	*RetrieveItem(ItemType &item, bool &bFound)*	*void*	*GetNextItem(ItemType &item)*

2. Write the **output** of the following 4 `cout` statements.

```
int   *p, *q;
int   x = 5;

p = &x;
q = p;
*q = 10;
*p = -4;

cout << p;
cout << q;
cout << *p;
cout << *q;
```

Programming

(13.5 Marks)

Fig. 1. *RollerCoaster Tycoon (RCT). RCT is a registered trademark of Chris Sawyer.*

1. Write a program to simulate the **world** of a **tile-based game**. In a tile-based game, such as *RollerCoaster Tycoon*, shown in Fig. 1, the game world, or *map*, is represented by a two-dimensional array of elements containing information about terrain, roads, objects, characters, enemies, etc. Each element, or *tile*, of the map is to contain a **terrain** type, either 0 for water or 1 for land, and an **original** (written by yourself) **unsorted list** in which all insertions take place at the **end** of the list, of objects, either 0 for a soldier, 1 for a monster or 2 for a vehicle. Note that a list of objects in a given tile may contain multiple instances of an object. For example, a list with the sequence 0, 1, 0, 2, 0 contains 3 soldiers, 1 monster and 1 vehicle. The map is to be represented by a 4×4 two-dimensional **array** of said tiles, encapsulated in a **class** *CMap*, with **member functions** to perform the functionally described below. Note that in each of the following, an individual tile is identified by its (*x*,*y*) position, where 0 ≤ *x*, *y* ≤ 3.

- A user can **get** and **set** the terrain type of a tile.

- A user can **get** the **number** of instances of an object in a tile.
- A user can **add** an object to a tile.
- A user can **remove 1 instance** of an object from a tile.
- A user can **remove all instances** of an object from a tile.

Your submission should include a screenshot of the program using the following script.

create a (4×4) *map*
set (all) tiles to land
set tile (1,2) *to water*
set tile (2,3) *to water*
print map to “map.txt” (output each tile in map; ex: “(2, 1): land; #S = 2, #M = 0, #V = 1*”)*

add soldier object to tile (3,2)
add soldier object to tile (3,2)
print (to screen) map tile (3,2) *(output as above; ex: “*(3,2): land; #S = 2, #M = 0, #V = 0*”)*
add monster object to tile (3,2)
print map tile (3,2)
remove 1 *soldier object from tile* (3,2)
print map tile (3,2)
remove 1 *monster object from tile* (3,2)
print map tile (3,2)
remove 1 *soldier object from tile* (3,2)
print map tile (3,2)

add vehicle object to tile (2,2)
add vehicle object to tile (2,2)
add vehicle object to tile (2,2)
add vehicle object to tile (2,2)
add vehicle object to tile (2,2)
add vehicle object to tile (2,2)
print map tile (2,2)
add monster object to tile (2,2)
add monster object to tile (2,2)
add monster object to tile (2,2)
print map tile (2,2)
remove 1 *vehicle object from tile* (2,2)
print map tile (2,2)
remove all monster objects from tile (2,2)
print map tile (2,2)
remove all vehicle objects from tile (2,2)
print map tile (2,2)