Task

Write C/C++ application containing the following three classes and the *main()* function with simple command prompt menu for testing and checking.

1. Class Date

Class *Date* is implemented by instructor. See files *Date.cpp* and *date.h* from Instructor's stuff.

2. Class Item

```
class Item
{
private:
	char Group;	// Any from range 'A'...'Z'
	int Subgroup;	// Any from range 0...99
	string Name;	// Any, but not empty
			// Alternative: string *pName;
			// By the student's choice
	Date Timestamp;	// Any
			// Alternative: Date *pTimestamp;
			// By the student's choice
public:
	Item(char, int, string, Date);
	.............	// To do
};
```

Class *Item* must have a constructor that fills the four fields above with random or pseudo-random values. Tips:

- To create a random date use method *Date::CreateRandomDate().*
- It is more comfortable to debug and check the results if the value of member *Name* is not a meaningless sequence of characters (something like *hsqgvxwqc*). Better take a text min 1000 words and using C standard function *strtok* split it into words. Insert the words into a vector, remove the duplicates and use a pseudo-random engine to get an index.

3. Class Data

Class *Data* must contain a C++ STL container containing objects of class *Item*. Its methods must allow the user to see the contents of container and to edit it. The students may select the type of container by their own discretion. Tip:

- The elements of a container may be items, pointers to items as well as some other containers.

Class Data must contain the following public methods:

1. *Data(int n);*
   Constructs the object and fills the container with *n* random items.
2. *~Data()*
   Destructs the object and releases all the memory occupied by the container and the items in it.

3. *void PrintAll();*
   Prints all the items stored in the container in command prompt window in easily readable format. Items from the same group and subgroup must be ordered by their names.
4. *void CountAll();*
   Returns the total number of items in the container.
5. *xxxx GetGroup(char g);*
   Returns a container or pointer to container of all the items from group *g*. The type of output value (here *xxx*) is the student's choice. The method must in some way (exception, *nullptr* on output, etc.) inform the user was the group found or not.
6. *bool PrintGroup(g);*
   Prints all the items from group g in command prompt window in easily readable format. Items from the same subgroup must be ordered by their names. Return value: *false* if the group does not exist.
7. *int CountGroup(g);*
   Returns the number of items in group *g*.
8. *xxxx GetSubgroup(char g, int sg);*
   Returns a container or pointer to container of all the items from group *g* and subgroup *sg*. The type of output value (here *xxx*) is the student's choice. The method must in some way (exception, *nullptr* on output, etc.) inform the user was the subgroup found or not.
9. *bool PrintSubgroupByNames(char g, int sg);*
   Prints all the items from group *g* and subgroup *sg* in command prompt window in easily readable format. Items must be ordered by their names. Return value: *false* if the subgroup does not exist.
10. *bool PrintSubgroupByDates(char g, int sg);*
    Prints all the items from group *g* and subgroup *sg* in command prompt window in easily readable format. Items must be ordered by their timestamps. Return value: *false* if the subgroup does not exist.
11. *int CountSubgroup(cgar g, int sg);*
    Returns the number if items from group *g* and subgroup *sg*.
12. *Item* GetItem(char g, int sg, string n);*
    Returns the pointer to the first of items specified by group *g*, subgroup *sg* and name *n*. If the item was not found returns *nullptr*.
13. *Item* GetItem(char g, int sg, Date d);*
    Returns the pointer to the first of items specified by group *g*, subgroup *sg* and timesatmp *d*. If the item was not found returns *nullptr*.
14. *bool PrintItem(char g, int sg, string n);*
    Prints the first of items specified by group *g*, subgroup *sg* and name *n* in command prompt window in easily readable format. If the item was not found returns *false*.
15. *bool PrintItem(char g, int sg, Date d);*
    Prints the first of items specified by group *g*, subgroup *sg* and timestamp *d* in command prompt window in easily readable format. If the item was not found returns *false*.
16. *bool RemoveGroup(char g);*
    Removes the group *g* and releases all the memory occupied by the items in it. If the group was not found returns *false*.

17. *bool RemoveSubgroup(char g, int sg);*
    Destructs the subgroup *sg* from group *g* and releases all the memory occupied by the items in it. If the group was not found returns *false*.
18. *bool RemoveItem(char g, int sg, string n);*
    Removes the first of items specified by group *g*, subgroup *sg* and name *n.* If the item was not found returns *false*.
19. *bool RemoveItem(char g, int sg, Date d);*
    Removes the first of items specified by group *g*, subgroup *sg* and timestamp *d*. If the item was not found returns *false*.

Requirement

Avoid to use *for* loops. Instead of that apply STL algorithms.