Data Structures: 1st Assignment Documentation

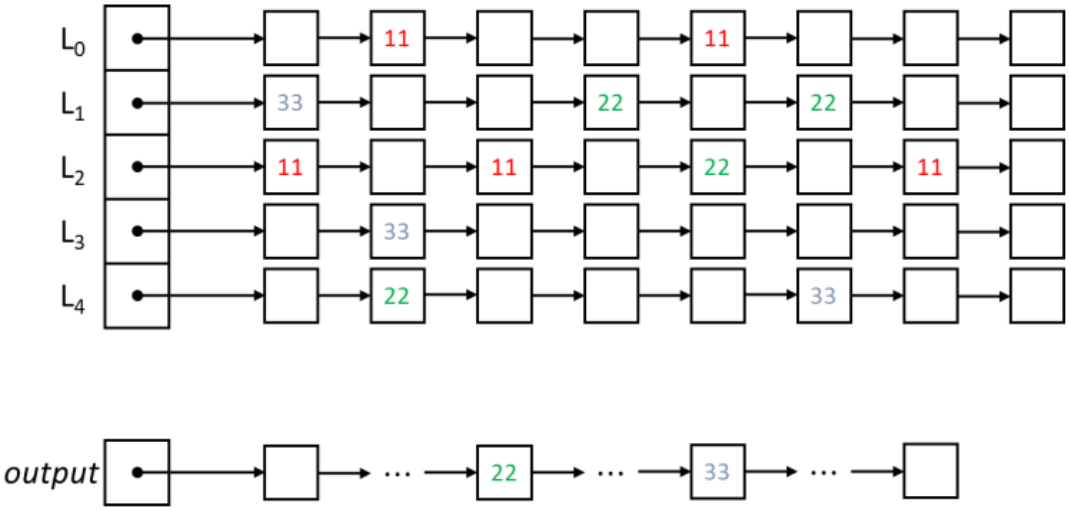
Project Participants:

- **Κριστιάν Γκολέμι Π18029,**
- **Χρήστος Μιχαήλ Καταγής Π18067**

Project Description:

Takes as input L linear lists and gives as an output a list that has the elements that are present at least at half of the L lists. The elements of the output list are sorted in ascending order and every one of them should appear only one time at the output list. The use of C++ Containers is prohibited.

Visual Example:



Code Analysis:

- **Libraries:**

  `#include <iostream>` : This library is used to support standard input-output functions such as: getting the number of lists from the user and showing him the output list.

  `#include <random>` : This library is used to help in the declaration of the size of each list, it is also used to initialize the random values for each list element .

`#include <ctime>` : This library is used for the time() function to provide a seed for the default_random_engine generator.

- **Classes:**

  The program consists of two classes: class `Chain` and class `ChainNode`. The two classes are friends, which means that any of the two can access each other's private and protected members.

  `Chain`: This class has a `Chain()` Constructor which initializes an empty chain list and `~Chain()` Destructor which deletes all the nodes in the chain list. The `Chain` class has also functionalities such as:

  Sort(const Chain<T>& x) which is a function that takes a whole chain list as a parameter and then taking into account each data value of each node, proceeds to sort the *nodes* in ascending order. This function is used only in the final output list.

  RemoveDupicates() which checks each node's data and compares it with every other node's data in the chain list in order to check if any of them have the same value. If that is true, the function proceeds to delete the duplicate nodes from the chain list completely. This function is used in the random generated lists at the start of the program and later on at the output list. The functionalities below are broadly explained at chapter 3.4 of the book "Data Structures, Algorithms and Applications in C ++" written by Sartaj Sahni.

  Find(int k, T& x), Search(const T& x), Delete(int k, T& x), Insert(int k, const T& x), Output(ostream& out)

  `ChainNode`: This class represents each node of a chain list having a variable named data which represents the integer hosted by the node, and a pointer *link which points to a ChainNode<T> element.

- **main:**

  The main function starts by asking the user for an integer that indicates the number of chain lists that he wants to be created. After that, the program proceeds to create these lists of random size, inserting in their nodes random values using the default_random_generator and the Insert() function. This occurs for each chain list and finally the lists are printed so that the user can see their initial form. Each list that is created is then stored in another chain list (the ChainList_List), which is a chain list that stores all the chain lists that

are created. This happens so that we can access the chain lists that are created in the for loop, outside it. After those actions, we finally have a list that contains all the chain lists that were created with each one of them having random size as well as random values stored in them. In the code (line 252), we have defined that the range of each lists size will be between 25 and 50 elements while the range of the values in each node will be between 0 and 50 (line 262) – these numbers can change defining any wanted range.

After creating the initial lists the algorithm in order to find the wanted elements and store them in the OutputList begins. First, the OutputList is created and then the function RemoveDuplicates() is called for each initial chain list, thus removing the duplicate elements from each chain list and finally printing the lists again, this time not having duplicate values. Later, using a number of for loops, we search for each element in all the lists – if this element exists in more than half of the lists, it's added to the OutputList. Initially, we define an outer loop that runs for the lists that are in the ChainList_List. For each list in the ChainList_List, we use the Find() function to find exactly the list and then through the inside loop we search each element of the current list. The current_element is found again by using the Find() function but this time in the current chain list. Afterwards, we search for this element in all the other lists that are inside the ChainList_List using two for loops. The first for loop searches if the current element exists in the previous lists, while the second for loop searches if the current element exists in the next lists saved in the ChainList_List. Every time that the current_element is found existing in another chain list, a variable occ_nr is increased by one. Finally, we check if the occ_nr for the current element is greater than half of the lists. If occ_nr > half of the lists number, it means that the current element exists in more than half of the lists, hence we add it to the OutputList. These actions take place for each element of each chain list through the for loops that are defined. Finally, the RemoveDuplicate() and Sort() functions are called again in order to sort and remove the duplicate values from the final OutputList, which is later on printed so that the user can see the final result.

Running the Program:

In order for the chainList.cpp file to be compiled on Windows, the user has to create a new command prompt session, move at the directory that the chainList.cpp file is stored and type the command below:

g++ chainList.cpp -o chainList

This will create an executable file named chainList.exe in the currently working directory for the user to run.

In order for this process to work the user needs to have installed the MinGW compiler and have the …/MinGW/bin directory added to the system variable PATH.

Execution Examples:

```
How many lists do you want to create? Type the number:
3
Creating 3 chain lists now!

These are the lists that were created without any changes:

List 1: 29 21 14 16 45 39 30 47 6 14 15 36 18 47 3 34 38 49 39 27 30 34 24 5 45 38 2 38 15 5
List 2: 44 47 22 35 36 1 9 1 44 45 39 24 26 17 27 36 6 4 28 45 43 24 45 42 16 16 48 41 28 17 29 41 32 13 2 11 38 12 30 30 5 35 27 37 15
List 3: 38 6 21 28 27 32 12 25 39 14 44 29 22 10 21 13 41 27 10 0 36 21 44 40 4 38 40 19 45 4 30 21 39 17

Now we are going to implement an algorithm in order to solve the problem!

These are the lists that were created after removing their duplicate elements:

List 1: 29 21 14 16 45 39 30 47 6 15 36 18 3 34 38 49 27 24 5 2
List 2: 44 47 22 35 36 1 9 45 39 24 26 17 27 6 4 28 43 42 16 48 41 29 32 13 2 11 38 12 30 5 37 15
List 3: 38 6 21 28 27 32 12 25 39 14 44 29 22 10 13 41 0 36 40 4 19 45 30 17

Final OutputList after removing its duplicate items and sorting it: 2 4 5 6 12 13 14 15 16 17 21 22 24 27 28 29 30 32 36 38 39 41 44 45 47
```

```
How many lists do you want to create? Type the number:
5
Creating 5 chain lists now!

These are the lists that were created without any changes:

List 1: 34 30 41 32 44 49 21 15 4 11 43 2 21 23 26 8 25 18 9 21 22 34 43 11 20 49 44 22 7 21 46 14 8 32 23 15 8 8 45 16 6 23 16 29
List 2: 20 7 38 22 16 41 9 20 6 39 39 2 1 8 36 25 32 38 23 50 0 32 8 30 31 5 14 41 50
List 3: 2 50 17 44 36 48 48 3 34 1 41 34 3 4 20 49 48 49 46 5 28 38 49 2 28 31 44 42 46 14 25 9 3 42 37 0 49 16 2 13 7 33 47 4
List 4: 26 28 2 33 34 22 27 47 14 40 5 30 34 24 19 30 20 44 6 5 19 31 12 40 26 43 32
List 5: 21 5 49 26 47 20 16 50 41 29 15 31 34 50 5 22 18 20 46 16 8 26 33 28 0 27 16 32 49 16 37 31 40 20 43 32 25

Now we are going to implement an algorithm in order to solve the problem!

These are the lists that were created after removing their duplicate elements:

List 1: 34 30 41 32 44 49 21 15 4 11 43 2 23 26 8 25 18 9 22 20 7 46 14 45 16 6 29
List 2: 20 7 38 22 16 41 9 6 39 2 1 8 36 25 32 23 50 0 30 31 5 14
List 3: 2 50 17 44 36 48 3 34 1 41 4 20 49 46 5 28 38 31 42 14 25 9 37 0 16 13 7 33 47
List 4: 26 28 2 33 34 22 27 47 14 40 5 30 24 19 20 44 6 31 12 43 32
List 5: 21 5 49 26 47 20 16 50 41 29 15 31 34 22 18 46 8 33 28 0 27 32 37 40 43 25

Final OutputList after removing its duplicate items and sorting it: 0 2 5 6 7 8 9 14 16 20 22 25 26 28 30 31 32 33 34 41 43 44 46 47 49 50
```

```
How many lists do you want to create? Type the number:
10
Creating 10 chain lists now!

These are the lists that were created without any changes:

List 1: 42 4 25 46 24 47 31 37 45 18 44 3 35 21 48 44 10 24 0 21 19 42 14 5 32 32 18 30 30 9 4
List 2: 17 42 17 19 32 0 40 23 18 17 34 11 32 33 18 36 5 50 50 38 14 27 22 27 2 7 30 18 6 5 22 29 7 43 37 29
List 3: 3 28 29 0 42 27 22 43 23 33 44 27 17 8 3 26 3 25 38 1 9 3 11 39 50 48 7 5 2 2 30 1 19 8 26 12 5 49 17 46 33 36 42 10 15
List 4: 4 2 17 21 8 25 31 40 39 36 46 17 36 48 1 38 24 43 32 20 47 30 28 11 38 21 17 48 45 26 26 22 33 36 0 4 31 5 4 15 26 44 40 17 17 35 5 32 50 24
List 5: 0 40 32 3 9 20 30 40 1 7 28 46 27 23 23 36 7 38 24 44 16 34 32 27 15 19 38 40 50 20 34 6 50 47 17 14 41 18 48 33 31 33 42 47 31 50 35
List 6: 24 11 16 33 19 14 38 6 36 15 12 39 27 22 42 15 12 45 0 23 0 23 13 17 18 15 42 41 5 31 23 0 15 36 48 39
List 7: 27 20 8 13 30 47 40 11 48 31 40 10 43 30 32 48 20 12 0 20 33 23 9 40 28 11 46 3 34 38 34 29 0 30
List 8: 2 16 22 30 25 35 29 37 20 37 26 37 33 7 25 33 47 20 28 31 47 10 46 47 26 33 23 20 3 46 44 0 1
List 9: 37 8 1 7 20 32 9 16 22 4 7 5 13 36 22 29 42 3 19 5 8 49 15 28 35 41 0 48 13 29 30 42 45 3 47 29 50 15 50 2 29 49
List 10: 5 48 26 42 10 47 40 0 6 25 45 42 25 29 30 35 30 0 40 0 24 43 27 26 38 20 9 17 43 0 19 20 25 33

Now we are going to implement an algorithm in order to solve the problem!

These are the lists that were created after removing their duplicate elements:

List 1: 42 4 25 46 24 47 31 37 45 18 44 3 35 21 48 10 0 19 14 5 32 30 9
List 2: 17 42 19 32 0 40 23 18 34 11 33 36 5 50 38 14 27 22 2 7 30 6 29 43 37
List 3: 3 28 29 0 42 27 22 43 23 33 44 17 8 26 25 38 1 9 11 39 50 48 7 5 2 30 19 12 49 46 36 10 15
List 4: 4 2 17 21 8 25 31 40 39 36 46 48 1 38 24 43 32 20 47 30 28 11 45 26 22 33 0 5 15 44 35 50
List 5: 0 40 32 3 9 20 30 1 7 28 46 27 23 36 38 24 44 16 34 15 19 50 6 47 17 14 41 18 48 33 31 42 35
List 6: 24 11 16 33 19 14 38 6 36 15 12 39 27 22 42 45 0 23 13 17 18 41 5 31 48
List 7: 27 20 8 13 30 47 40 11 48 31 10 43 32 12 0 33 23 9 28 46 3 34 38 29
List 8: 2 16 22 30 25 35 29 37 20 26 33 7 47 28 31 10 46 23 3 44 0 1
List 9: 37 8 1 7 20 32 9 16 22 4 5 13 36 29 42 3 19 49 15 28 35 41 0 48 30 45 47 50 2
List 10: 5 48 26 42 10 47 40 0 6 25 45 29 30 35 24 43 27 38 20 9 17 19 33

Final OutputList after removing its duplicate items and sorting it: 0 3 5 9 17 19 20 22 23 27 28 29 30 31 32 33 35 36 38 42 46 47 48
```

Contact:

- golemikristian14@gmail.com
- 0713549@gmail.com