Linköping Institute of Technology
Campus Norrköping
Department of Science and Technology/ITN
Aida Nordman

Exam for TNG033

January 7th, 2013, 08.00 am - 12.00 pm

Aid: Any book, solutions for exercises, and access to web pages. However, the **exam is individual**. Thus, you are not allowed to use e-mail. If you copy any code from a web page then you must explicitly indicate that fact and write the web page address in a comment line preceding the copied code. TNG033 course webpage requires a **`login = TNG033`** and a **`password = TNG033ht13`**.

Files needed: Files needed for this exam are available in the folder **`D:\TENTA\TNG033`**.

Instructions: Use a computer in the lab room to log in with your username and password. Then, start **`Code::Blocks`**. **Private laptops are not allowed in the exam**.

You must **write your name and personal number in the beginning of every file** (i.e. **`.cpp`**, **`.h`**), in a comment line.

Do not turn off the computer any time. Only log out when you have finished the exam.

To simplify the marking process, for each exercise, all code is given in the same file. Thus, **deliver one source file for each exercise**.

Delivering your solutions: In the folder **`D:\TENTA\TNG033`** create a sub-folder for each exercise, called **`Exerc1`**, **`Exerc2`**, *osv*.

Save your source code files (e.g. **`.cpp`**, **`.h`**) for exercise **`1`** in the folder **`D:\TENTA\TNG033\Exerc1`**. For exercise **`2`**, you save the source files in the folder **`D:\TENTA\TNG033\Exerc2`**, *osv*.

Use your LiU student e-mail address and rename the folder **`TNG033`** in **`D:\TENTA`** with it. For example, if your LiU e-mail address is **`magsi007`** then the folder name **`D:\TENTA\TNG033`** should become **`D:\TENTA\magsi007`**.

Results: Results are e-mailed automatically to every student from LADOK.

Good luck!

Exercise 1 [35 p]

In the file **ex1.cpp**, you can find the definition of a (simple) class **List** to represent a singly linked list such that each node stores an integer. Notice that the list may contain repeated values and it may not be sorted. The only (member) functions available are a constructor, a destructor, a copy constructor, and an **operator<<**. No dummy nodes are used in the implementation of the lists. Add to the class the functionality described below.

a. A function **which_order** that determines whether the nodes of the list are sorted increasingly, decreasingly, or are equal, or neither of the previous cases. Then, the function returns **1**, **-1**, **0**, or **2**, respectively.

b. A function **list_interval_values** that, given a list **L1** and two integer **i** and **j**, returns a new list **L2** such that any value **i** ≤ **v** ≤ **j** in **L1** must be stored in a node of **L2**. If there are no values stored in **L1** that belong to the interval **[i,j]** then **list_interval_values** should return an empty list. For instance, if
L1 = 1 → 4 → 3 → 2 → 4 → 7
Then, **L1.list_interval_values(0,4)** returns the list
L2 = 1 → 4 → 3 → 2 → 4

Add your code to the file **ex1.cpp** that also contains a **main()** function to test your code. The **main()** function cannot be changed. A possible output can be found in the file **ex1_out.txt**.

Exercise 2 [30 p]

Write a program that starts by reading a text file storing information about movies: movie's name (e.g. "*Blade Runner*") and two integer scores, for each movie. The file consists of a sequence of lines such that each (valid) line has one of the formats below, either

```
movie_name score1 score2
```
or
```
score1 score2 movie_name
```

Assume that each movie occurs only once in the file.

The program then performs, step by step, the actions described below (a. to g.). Notice that you should use the **Standard Template Library** (**STL**), whenever possible. To get full points, it is also required to use standard algorithms, instead of hand-written loops (such as **`for`**-loop, **`while`**-loop, or **`do`**-loop), if possible.

Add your code to the file **`ex2.cpp`**. You can write the corresponding code directly in the **`main`** function, as indicated in **`ex2.cpp`**, and do not delete the comments given in the **`main`**.

The file **`data.txt`** can be used to test your program and the expected output is available in the file **`ex2_out.txt`**.

a. Request to the user the name of the input text file and open it. If opening the file does not succeed then the program should print an error message and terminate.

b. Declare a new type **`Table`** representing a **`map`** such that the key is a string (movie name) and the associated value are the movie scores.

c. Load the file into a variable **`T`** of type **`Table`**. Note that movies names stored in **`T`** should be converted to upper case letters (e.g. "*Blade Runner*" is converted to "*BLADE RUNNER*"). The file may be garbled and any lines not as described above should be simply ignored.

d. Display, to **`cout`**, the number of movies stored in table **`T`**.

e. Display, to **`cout`**, all the information stored in table **`T`** (i.e. movie names and corresponding scores) sorted alphabetically by movie name.

f. Display, to **`cout`**, all the information stored in table **`T`** (i.e. movie names and corresponding scores) sorted increasingly by first score. Movies that have the same first score should then be sorted increasingly by the second score.

g. Request to the user the name of a movie and the corresponding couple of scores. Then, add the new movie to table **`T`**, if the given movie does not exist in **`T`**. Finally, display again, to **`cout`**, all the information stored in table **`T`**.

Exercise 3 [35 p]

Design a class hierarchy for representing different characters of a game. All characters are are **A**dvanced **A**vatars (**AA**). There are two types of advanced avatars, **B**eautiful **B**easts (**BB**) and **C**urious **C**heaters (**CC**). Advanced avatars may have friends who are also advanced avatars and some avatars are shy.

AA shall be base class for all types of avatars. This class should store the *name* (**string**) of an avatar and the *list of his friends*. Moreover, the class should have the following member functions.

- A function **is_shy()** that returns a boolean indicating whether an avatar is shy.
- A function **talk** that advanced avatars use to present themselves. This function should display the avatar's name.
- An **operator+=** to add an advanced avatar to the list of friends of another avatar. For instance, if **b1** and **b2** are **AA**s then **b1 += b2;** adds **b2** to the list of friends of **b1**.
- An **operator-=** to remove the avatar with a given name from the list of friends of another advanced avatar. For instance, if **b** is an **AA** and **s** is a **string** then **b -= s;** removes avatar named **s** from the list of friends of **b**.

BB is a direct subclass of **AA**. When a new **BB** is created, it should be possible to define whether the avatar is shy, and thereafter it should not be possible to change his shyness. If no information is given about shyness then it is assumed that the avatar is not shy. When a no shy **BB** avatar talks (i.e. member function **talk** is called), he says how many friends has (besides his name) and should also let his friends talk and introduce themselves.

CC is a direct subclass of **AA** and **CC** avatars are always shy.

Note that your class hierarchy should also satisfy the following conditions.

- It should not be possible to create **AA** objects that are not instances of one of the **AA** sub-classes (such as **BB** or **CC**).
- It should be possible to keep track of the number of advanced avatars without any friends.
- It should be possible to extend the class hierarchy with new types of advanced avatars, i.e. new sub-classes of **AA** can be added in the future.

Design your classes with care and keep with the given specifications.

Add your code to the file **ex3.cpp** that also contains a **main()** function to test your code. Add to the **main** function the code needed to display the number of advanced avatars without any friends. The **main()** function cannot be changed, otherwise. The expected output is available in the file **ex3_out.txt.**