TCP1201 Objected-Oriented Programming and Data Structures
Assignment 2

Trimester 2, Session 2014/2015
Faculty of Computing and Informatics
Multimedia University

DUE DATE: 21 Jan 2015 (Wed), 11:59pm

Outline

This assignment contributes 10% of the total course marks. This assignment consists of only one question. Every student is to submit one assignment individually.

You are strongly advised to submit the assignment on time even though it is incomplete. Late submissions will be given a zero mark. All assignments (across both TC01 and TC02) will be subject to a plagiarism checker, and those found to have plagiarized from other students' works or other sources will also be given a zero mark.

If necessary, selected students will be called for an oral interview by your respective lecturer. The date and time of interview will be scheduled and announced after the assignment deadline.

Make sure your program code can be compiled under g++ 4.7.1. You may download Code::Blocks 13.12 with bundled g++ 4.8.1 at http://sourceforge.net/projects/codeblocks/files/Binaries/13.12/Windows/codeblocks-13.12mingw-setup.exe

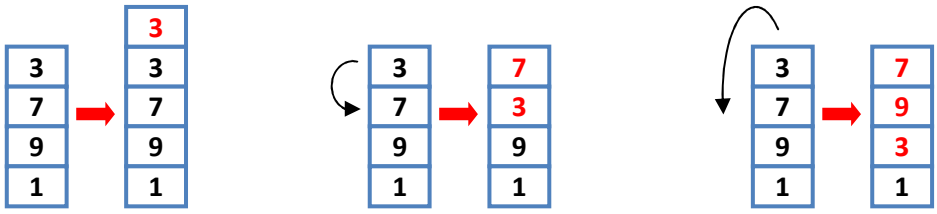
Problem Statement

E-leisure.com is undergoing an upgrade to test out some new features. In particular, a text file will be used to store data statistics of all the books sold through the online store. In order to achieve this, there are 2 main tasks in this assignment that you are to attempt.

1) XStack

Create a template pointer-based stack class called **XStack** based on the standard stack ADT (push, pop, peek, isEmpty, makeEmpty methods) and extend it to incorporate the following new methods:

a) **duplicate –** Make another copy of the top item in the stack and push it back to the stack
b) **swapTop –** Exchange the position of the two top-most items on the stack
c) **roll –** Rotate the position of the *n* top-most items by one step upwards. For example, if *n*=3, items at positions 1-2-3 will be moved to positions 2-3-1 respectively.



duplicate **swapTop** **roll** (n=3)

For this task, write a simple demo program that is able to demonstrate that these three new operations work as required. You may use simple data types such as integers or strings.

Note:

- Please do proper separation of files (header file, implementation file and the main application file)
- You may add other supporting (private) methods to XStack if needed.

2) Book Statistics

Processing statistics from data is an important task for their business. In this task, you are to extract some important statistics from book data stored in a text file database. You are to use the provided class **BookItem** (.hpp and .cpp files), together with the **XStack** class that you have created in task 1, to create the following features and statistics:

a) **Book search –** Simple search feature that takes in a book serial number and returns other information of the searched book. You should return no matches, if the book does not exist.
b) **Top 5 best sellers** **–** Top 5 books that sold the most, in descending order of copies sold
c) **Bottom 5 worst sellers –** Bottom 5 books that sold the least, in ascending order of copies sold
d) **10 newest books –** Most recent books, according to year published. If there is a tie, follow alphabetical order then.

In the sample application file (*main.cpp*) provided, the STL Stack class is used to demonstrate how a stack is used in this program. Of course, you are required to replace this class with the **XStack** class that you have created in task 1.

readData() reads the database text file (*db_small.txt* consisting of only 30 entries), displays rows of data that have been read, and then stores the data into the stack *stBooks.* A few lines of sample code following that shows how the stack *stBooks* can be utilised.

The output after compiling the sample application file is shown below:

```
Reading data...
 14833 |  Alice in Wonderland |  Lewis Carroll |  1865 | 28
 15322 |  War of the World |  H.G. Wells |  1953 | 17
 20983 |  The City |  Dean Kootnz |  2014 | 36
 19123 |  The Hobbitt |  J.R.R. Tolkien |  1938 | 178
 24504 |  The Old Man and the Sea |  Ernest Hemingway |  1951 | 135
 13323 |  My Experiments with Truth |  Mahatma M.K.Gandhi |  1929 | 6
 19232 |  Far from the Madding Crowd |  Thomas Hardy |  1874 | 2
 11219 |  The Merchant of Venice |  William Shakespeare |  1605 | 26
 15677 |  A Tale of Two Cities |  Charles Dickens |  1859 | 37
 15899 |  David Copperfield |  Charles Dickens |  1850 | 29
 13298 |  Gulliver's Travels |  Jonathan Swift |  1726 | 67
 17666 |  Pride and Prejudice |  Jane Austen |  1813 | 82
 11889 |  Time Machine |  H.G. Wells |  1895 | 13
 20981 |  Mein Kampf |  Adolf Hitler |  1925 | 10
 15245 |  Invisible Man |  H.G. Wells |  1897 | 16
 12290 |  Around the World in Eighty Days |  Jules Verne |  1873 | 58
 23280 |  Nineteen Eighty-Four |  George Orwell |  1949 | 103
 19911 |  War and Peace |  Leo Tolstoy |  1869 | 149
 14923 |  Adventures of Sherlock Holmes |  Arthur Conan Doyle |  1892 | 66
 19002 |  Robinson Crusoe |  Daniel Defoe |  1719 | 24
 11198 |  Faust |  Goethe |  1808 | 0
 18725 |  A Doctor in the House |  Mahathir Mohamad |  2011 | 26
 15533 |  The Singapore Story |  Lee Kuan Yew |  1998 | 20
 18929 |  A Brief History of Time |  Stephen Hawking |  1998 | 25
 17911 |  Les Miserable |  Victor Hugo |  1862 | 96
 24501 |  The Bourne Identity |  Robert Ludlum |  1980 | 71
 14796 |  The Lion the Witch and the Wardrobe |  C.S. Lewis |  1950 | 68
 18989 |  The Mist |  Stephen King |  1980 | 9
 15263 |  Cosmos |  Carl Sagan |  1980 | 20
 18677 |  Anna Karenina |  Leo Tolstoy |  1877 | 136
30 row(s) read.

From top of stack:
<18677> Anna Karenina : Leo Tolstoy (1877) - 136 copies sold
<15263> Cosmos : Carl Sagan (1980) - 20 copies sold
```

Note:

- You are free to modify the codes given (BookItem and main application file) to suit your program. Important: You do not need to understand how readData() works, so please do not tamper with it if you are not familiar!
- This assignment does not require you to perform any update to the text file database.
- You do not need the three new methods in task 1 to be able to do task 2. However, they *may* be useful in some way. Feel free to add other methods that will help you in task 2.
- Hard-coding the answer to these statistics is not allowed! Anyway, the instructors will be using a different set of book data during evaluation. Moreover, we will provide a larger set of data a few days before the deadline to help you test your program further ☺.

Database text file

```
14833,Alice in Wonderland,Lewis Carroll,1865,28
15322,War of the World,H.G. Wells,1953,17
20983,The City,Dean Kootnz,2014,36
19123,The Hobbitt,J.R.R. Tolkien,1938,178
24504,The Old Man and the Sea,Ernest Hemingway,1951,135
13323,My Experiments with Truth,Mahatma M.K.Gandhi,1929,6
19232,Far from the Madding Crowd,Thomas Hardy,1874,2
11219,The Merchant of Venice,William Shakespeare,1605,26
```

'*db_small.txt*' contains information of the books that have been sold previously. Each row consists of the following book data, delimited by a comma (,):

```
serial number, book title, author name, year published, copies sold
```

If you are compiling at command prompt, this file should be placed in the same directory as your codes. However, if you are creating a CodeBlocks project, the database file should be placed in the same location as the project file (.cbp).

What to hand up

Although there are two separate tasks in this assignment, you can encouraged to combine both tasks into a single program (either showing one followed by the other, OR create a simple selection menu to provide an option of which to show)

Bonuses (Optional)

These are more challenging features that you can attempt IF you have finished the required tasks above. Additional bonus marks (up to 3% but not exceeding your total coursework marks) will be awarded if you can do the following:

- **Top *n* unique authors –** Show the top *n* book unique (unrepeated) authors. Notice that some authors have multiple books. To find this, you should not total up books from the same authors (unfair!) as not all authors have the same number of books in the store. Instead, sort the books according to the copies sold, and take only the top unique authors.
- **Keyword search** – Create a search books by a keyword of the title. For example, searching for the keyword "world" should return to the user, two matched books: "War of the World" by H.G. Wells and "Around the World in Eighty Days" by Jules Verne.

Submission Format

- Create a zip file name

*TT0**X**_AS2_**StudentID**_**StudentName***.zip

Modify the parts in red. *TT0X* is your **lab section** (replace the 'X' with the correct number). *StudentID* is your student ID. *StudentName* is your name. RAR or 7ZIP files are also acceptable. The zip should contain

a) **Source code files** (*.hpp and *.cpp) or **complete set of project files** (*.cbp, *.hpp, *.cpp, if you have used CodeBlocks Project). Do not attach any .exe file as some mail servers such as gmail would reject it. One easy way is to delete away the generated .exe files before zipping up for submission.
b) **User manual** (*.txt), containing simple instructions outlining how to use your program, and what your program can do.

2. Compose an email with subject as "*TT0X AS2 StudentID StudentName*".
3. Attached the *TT0X_AS2_StudentID_StudentName.zip* created in Step 1.
4. Send the email with the attachment to your lecturer's MMU email.
5. The submission will be regarded as late according to email timestamps, if it reaches the lecturer's mail box after the deadline. Email at least several hours earlier to be safe, in case you encounter any potential Internet problems at the wrong timing. Do not make multiple submissions. Check carefully before submitting.

Feature Sheet & Evaluation Criteria

Criteria	Item
1. XStack (4 marks)	1.1. Duplicate [1m]
	1.2. SwapTop [1]
	1.3. Roll [1.5m]
	1.4. Basic Stack ADT complete [0.5m]
2. Book Statistics (5 marks)	2.1. Successfully integrated XStack [0.5m]
	2.2. Book search [1.5m]
	2.3 Top 5 best sellers [1m]
	2.4 Bottom 5 worst sellers [1m]
	2.5 10 newest books [1m]
3. Miscellaneous (1 mark)	3.1 Good programming style, proper indentation [0.5m] 3.2 User manual attached [0.5m]
4. Bonus* [max. 3 marks to coursework mark]	Top *n* unique authors, Keyword search (see details earlier)
5. Interview** (0 mark for the assignment if fail to be present for interview)	Fluency in using the program
	Ability to explain code
6. Work found to have plagiarized from another source, or Late submission	0 mark for the whole assignment

Note:

***** Bonuses are added on top of your total assignment mark of 10, and will be awarded based on the discretion of your lecturer. However, total coursework marks will be capped at 60 marks.

** Interview carries no mark (hence, no difference for students who are called for interview and those who are not). It is for the lecturers to ask questions if there are any doubts over your assignment, or if you are required to clarify parts of your work so that the lecturers understand your work better. However, it is compulsory to be present for interview if selected.