CSY2006: Software Engineering 2

Due for Issue (week commencing):	**Monday, 30th March 2020**	**Date for Submission:**	**Sunday, 03rd May 2020 23:59**
		Module Tutor:	Dr Suraj Ajit

This assignment is weighted as 50% of the Module's assessment

Assessment Rubrics

Aspect (& weighting)	Excellent A	Good B	Satisfactory C	Needs some more work D	Needs much more work F
Task Solution (Working Code) (60%)					
Code quality, Error handling and Recovery (15%)					
Testing (15%)					
Documentation/Comments/ Code Convention (10%)					

The University of Northampton Policy on Plagiarism & Mitigating Circumstances will be strictly implemented. By submitting this assignment you are asserting that this submission is entirely your own/individual work.

CSY2006 Software Engineering 2
Assignment

Brief:

Implement software solutions for the following tasks using the C++ programming language. Use the standard C++ code convention and document the code with relevant comments. Test the program with reasonable number of test cases.

Tasks:

1. Design a class called Date. The class should store a date in three integers: month, day and year. There should be member functions to print the date in the following forms:
   12/25/2012
   December 25, 2012
   25 December 2012
   Input Validation: Do not accept values for the day greater than 31 or less than 1. Do not accept values for the month greater than 12 or less than 1.
   Demonstrate the class by writing a main program to implement it. Save the main program as `DateDemo.cpp`. (10 marks)

2. Extend/modify the above Date class to contain the following overloaded operators:
   ++ Prefix and Postfix increment operators: These operators should increment the object's day member.
   -- Prefix and Postfix decrement operators: These operators should decrement the object's day member.
   - Subtraction operator: If one date object is subtracted from another, the operator should give the number of days between the two dates. For example, if April 10, 2012 is subtracted from April 18, 2012, the result will be 8.
   << cout's stream insertion operator: This operator should cause the date to be displayed in the form April 18, 2012
   >> cin's stream insertion operator: This operator should prompt the user for a date to be stored in a Date object.
   The class should detect the following conditions and handle them accordingly:
   - When a date is set to the last day of the month and incremented, it should become the first day of the following month.
   - When a date is set to December 31 and incremented, it should become January 1 of the following year.
   - When a day is set to the first day of the month and decremented, it should become the last day of the previous month.
   - When a date is set to January 1 and decremented, it should become December 31 of the previous year.
     Demonstrate the class's capabilities in a simple program. Save the program as `AdvancedDateDemo.cpp`.
     Input Validation: the overloaded >> operator should not accept invalid dates. For example, the date 13/45/2012 should not be accepted. (20 marks)

3. Extend/modify the above Date class so that it implements the following exception classes:
   InvalidDay - Throw when an invalid day (<1 or >31) is passed to the class.
   InvalidMonth - Throw when an invalid month (<1 or >12) is passed to the class.
   Demonstrate the class in a driver program. Save the program as `ExceptionDateDemo.cpp`. (10 marks)

4. Design a `Ship` class that has the following members:
   - A member variable for the name of the ship
   - A member variable for the year that the ship was built
   - A constructor and appropriate accessors and mutators

- A virtual print function that displays the ship's name and the year it was built

Design a `CruiseShip` class that is derived from the `Ship` class. The `CruiseShip` class should have the following members:

- A member variable for the maximum number of passengers
- A constructor and appropriate accessors and mutators
- A print function that overrides the print function in the base class. The `CruiseShip` class's print function should display only the ship's name and the maximum number of passengers.

Design a `CargoShip` class that is derived from the Ship class. The `CargoShip` class should have the following members:

- A member variable for the cargo capacity in tonnage (an int)
- A constructor and appropriate accessors and mutators.
- A print function that overrides the print function in base class. The `CargoShip` class's print function should display only the ship's name and the ship's cargo capacity.

Demonstrate the classes in a program that has an array of `Ship` pointers. The array elements should be initialized with the addresses of dynamically allocated `Ship, CruiseShip, and CargoShip` objects. The program should then step through the array, calling each object's print function. Save the main program as `ShipDemo.cpp`. (15 marks)

5. Design a class `NumDays`. The class's purpose is to store a value that represents a number of work hours and convert it to a number of days. For example, 8 hours would be converted to 1 day, 12 hours would be converted to 1.5 days, and 18 hours would be converted to 2.25 days. The class should have a constructor that accepts a number of hours, as well as member functions for storing and retrieving the hours and days. The class should also have the following overloaded operators:
   - + Addition operator: When two `NumDays` objects are added together, the overloaded + operator should return the sum of the two objects' hours members.
   - Subtraction operator. When one `NumDays` object is subtracted from another, the overloaded – operator should return the difference of the two objects' hours members
   - ++ Prefix and postfix increment operators. These operators should increment the number of hours stored in the object. When incremented, the number of days should be automatically recalculated.
   - Prefix and postfix decrement operators. These operators should decrement the number of hours stored in the object. When decremented, the number of says should be automatically recalculated.

   Demonstrate the class in a program. Write specification (.h) and implementation (.cpp) files for the `NumDays` class. Save the main program as `NumDaysDemo.cpp`. (20 marks)

6. A system to store the internally awarded and externally awarded exam marks of a student is required. Design three classes `InternalExamMarks`, `ExternalExamMarks` and `StudentMarksInfo`. The class `StudentMarksInfo` inherits both the classes `InternalExamMarks` and `ExternalExamMarks`. The student class has data members such as roll number and stream. The `InternalExamMarks` class stores the internally awarded exam marks for two subjects and the `ExternalExamMarks` class stores the externally awarded exam marks for two subjects. Write a program to model this relationship by using appropriate member functions and demonstrate multiple inheritance. Save the main program as `Multiple_Student.cpp`. (10 marks)

7. Design a linked list class (`StringList`) to hold a list of strings. Include the usual member functions to append, insert and delete nodes. In addition, implement the following member functions. Function headers are listed below (string comparisons need to be case sensitive).

   a) //delete node at specified position. A value of 0 means first node is deleted.
   void removeByPosition(String pos)
   b) //add the elements in otherList to this list (no duplicates)
   void addAll(const StringList &otherList)

c) //remove all the elements in otherList from this list
void removeAll(const StringList &otherList)

d) //retain the elements in this if they are also in otherList
void retainAll(const StringList &otherList)

Demonstrate and test the class using sample values. Save the main program as `StrListDemo.cpp`. (20 marks)

8. Create a linked list class template with the usual member functions (append, inset, delete, display). Include a function to overload operator []. This will give the linked list the ability to access nodes using a subscript, like an array. The subscript 0 will reference the first node in the list. If an invalid subscript is used, the function should throw an exception. Save the main program as `OverLoadedLL.cpp`.

   (10 marks)

9. A string may use more than one type of delimiter to bracket information into "blocks". For example, a string may use braces { } and parentheses ( ) as delimiters. A string is properly delimited if each right delimiter is matched with a preceding left delimiter of the same type in such a way that either the resulting blocks of information are disjoint, or one of them is completely nested within the other. Write a program that uses a **single stack** to check whether a string containing braces, parentheses, and brackets is properly delimited. (10 marks)

   For example:

```
[ { ( ) } ]        -- correct
{ ( { } ) () [] } -- correct
{ [ ( } ) ]        -- incorrect
[ ( ( { ) } ] )    -- incorrect
```

   For extra credit (+5 marks), extend the program so that any delimiters appearing inside double quotes are ignored for meaningful matching check. For example, " :-) " is correct.
   Save the main program as `BalancedDelimiterCheck.cpp`.

10. Write the definition of the function `moveNthFront` that takes as a parameter a positive integer n. The function moves the nth element of the queue to the front. The order of the remaining elements remains unchanged. For example, suppose:

    Queue = {5, 11, 34, 67, 43, 55} and n=3

    After a call to the function `moveNthFront`:

    Queue = {34, 5, 11, 67, 43, 55}.

    (i) Implement the above function for a Static Queue Template class. (10 marks)

    (ii) Implement the above function for a Dynamic Queue Template class (5 marks)

    Save the main program as `QueueMoveOperations.cpp`.

11. Implement the following two functions:

    - A function named `downsize` that accepts a string linked list as a reference parameter and removes every second value from the linked list
    - A function named `merge` that accepts two string linked lists as reference parameters and merges them into one linked list, alternating elements from each list until the end of one of the lists has been reached, then appending the remaining elements of the other list. For example, merging the lists containing A B C and D E F G H should yield the list A D B E C F G H.

    Demonstrate the above functions in a program. Save the main program as `LinkedListOps.cpp`.

(10 marks)

12. Create a binary search tree and write a function `onlyChild` that returns the number of nodes in a binary tree that have only one child. Demonstrate the above function in a program. Save the main program as `BinaryTreeOnlyChildDemo.cpp`. (10 marks)

13. Write a program to do the following:
    a. Build a binary search tree T_1
    b. Do a post order traversal of T_1 and while doing the post order traversal, insert the nodes into a second binary search tree T_2
    c. Do a preorder traversal of T_2 and while doing the preorder traversal, insert the node into a third binary search tree T_3
    d. Do an inorder traversal of T_3
    e. Output the heights and the number of leafs in each of the three binary search trees

    Save the main program as `BinarySearchTreeOps.cpp` (10 marks)

Deliverables/Submission:

All requirements below **MUST** be delivered to achieve a **passing grade** for this assignment.

- E-Submission of one **WORD document** containing full **source code listing** of each task together with the corresponding output generated for different sets of input (in the form of **screenshots** of console output) through Turnitin on NILE. To do this, go to the NILE site for this module and use the link labelled 'Submit your work'.

- E-Submission of link to **video**. You are recommended to use Kaltura for video. Ensure that the video link is publicly accessible. The video should be no longer than 20 minutes. Your **face** and **voice** should be clear in the video. In the video, you should run each program, show the output generated and also provide brief explanation about the code. To submit the link, go to the NILE site for this module and use the link labelled 'Submit your work'.

- E-Submission of a single ZIP archive that contains all the **source code files** (.cpp).
  Save the source code files to each task in a separate folder that is named with the corresponding task number, e.g. the folder named Task1 would contain a file StarTriangle.cpp. Then save all the solution folders in another folder, which must be zipped and named with your student ID, prior to uploading to the NILE system, e.g. *12345678.zip* where 12345678 is your student ID. To do this, go to to the NILE site for this module and use the link labelled 'Submit your work'. Clicking on the link (*SourceCodeEsubmission*), will take you into the submission form, where you can upload your ZIP archive using the 'Attach File' button (*Browse for Local File*). Finally, click the *Submit* button.

Assessment Guidelines:

1. Please note that the work you submit must be your own and you **may be asked for a viva** over collaborate/skype.
2. Evidence of Testing/Error Handling for each task needs to be demonstrated by providing screenshots of console output for different sets of inputs (including erroneous inputs).
3. Strict penalties will be applied if you do not follow the assignment specification (e.g. method name, program/file name has to be exactly as stated in the question). The video link should be made publicly accessible.