OOP 244 – Introduction to Object

Oriented Programming

Assignment 02

**Q1 – String Manipulator** (25 marks)

Write a program to manipulate strings. In this program take a whole paragraph with punctuations (up to 500 letters) either input from user, initialize or read from file and provide following functionalities within a class:

1. Declare class Paragraph\_Analysis
2. Member Function: SearchWord (to search for a particular word)
3. Member Function: SearchLetter (to search for a particular letter)
4. Member Function: WordCount (to count total words)
5. Member Function: LetterCount (ONLY to count all letters e.g ‘A’,’a’)
6. Member Function: FindReplaceWord (to find and replace a word)
7. Member Function: FindReplaceLetter (to find and replace a letter)
8. Member Function: Summary (to display summary of frequency of each letter within the paragraph)
9. Of course, a menu is expected by user to know about available functionalities of your Paragraph\_Analysis application.

Note: you can use access specifiers, return types and list of parameters of your own choice.

**Q2 – Search and Sort Integer Array** (25 marks)

Write a program that prompts the user to enter the number of elements and the numbers themselves to be placed in an integer array that holds a maximum of elements specified by end user. The program should then prompt the user for an integer which will be searched for in the array using a binary search. Make sure to include the following steps along the way:

1. Ask the user how many elements would be entered. An array of integers with this many element should then be dynamically allocated.
2. A sort routine must be called before the binary search. You may use either the selection sort or the bubble sort. However, the sort must be implemented in its own function and not in main.
3. Next include a function called by main to implement the binary search.
4. The ordered array produced by the sort should be passed to the search routine which returns the location in the sorted array of the sought value, or -1 if the value is not in the array.
5. Add values returning function that computes the average, median, and mode mean of your data set.
6. Recall that the mean is the sum of the data values divided by the number of pieces of data. Your program should output the size of the array entered, the array as entered by the user, the sorted array, the integer being searched for, the location of that integer in the sorted array (or an appropriate message if it is not in the array), and the mean of the data set.
7. Modify your program so that the data is kept in a vector instead of an array. Functionalities and calculations will be same as array solution, but you will change your solution with vector. Your implementation with vector should be in a separate file.

Note 1: Do not accept negative numbers for input

Note 2: Use pointers and dynamic memory allocation

**Q3 – Account Inheritance Hierarchy** (50 marks)

Create an inheritance hierarchy that a bank might use to represent customers’ bank accounts. All customers at this bank can deposit (i.e., credit) money into their accounts and withdraw (i.e., debit) money from their accounts. More specific types of accounts also exist. *Savings* accounts, for instance, earn interest on the money they hold. *Checking* accounts, on the other hand, charge a fee per transaction (i.e., credit or debit).

Create an inheritance hierarchy containing base class *Account* and derived classes *SavingsAccount* and *CheckingAccount* that inherit from class Account. Base class Account should include one data member of type double to represent the account balance. The class should provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the initial balance to ensure that it’s greater than or equal to 0.0. If not, the balance should be set to 0.0 and the constructor should display an error message, indicating that the initial balance was invalid. The class should provide three member functions. Member function credit should add an amount to the current balance. Member function debit should withdraw money from the Account and ensure that the debit amount does not exceed the Account’s balance. If it does, the balance should be left unchanged and the function should print the message "Debit amount exceeded account balance." Member function getBalance should return the current balance.

Derived class SavingsAccount should inherit the functionality of an Account, but also include a data member of type double indicating the interest rate (percentage) assigned to the Account. SavingsAccount’s constructor should receive the initial balance, as well as an initial value for the SavingsAccount’s interest rate. SavingsAccount should provide a public member function calculateInterest that returns a double indicating the amount of interest earned by an account. Member function calculateInterest should determine this amount by multiplying the interest rate by the account balance. [Note: SavingsAccount should inherit member functions credit and debit as is without redefining them.]

Derived class CheckingAccount should inherit from base class Account and include an additional data member of type double that represents the fee charged per transaction. CheckingAccount’s constructor should receive the initial balance, as well as a parameter indicating a fee amount. Class CheckingAccount should redefine member functions credit and debit so that they subtract the fee from the account balance whenever either transaction is performed successfully. CheckingAccount’s versions of these functions should invoke the base-class Account version to perform the updates to an account balance. CheckingAccount’s debit function should charge a fee only if money is actually withdrawn (i.e., the debit amount does not exceed the account balance).

[Hint: Define Account’s debit function so that it returns a bool indicating whether money was withdrawn. Then use the return value to determine whether a fee should be charged.]

After defining the classes in this hierarchy, write a program that creates objects of each class and tests their member functions. Add interest to the SavingsAccount object by first invoking its calculateInterest function, then passing the returned interest amount to the object’s credit function.

Note: UML diagrams of the classes should be provided separately.

**Notes:**

* Total mark is 100.
* Use MS Visual C++ or Eclipse while answering questions. If you don't have any of these IDEs installed on your home computer, I suggest that you install one of them first.
* You can follow the documents that are on the Blackboard to install these IDEs. It is located under Course Documents -> How to Documents.
* If you use any other IDE other than MS Visual C++ or Eclipse, you must make sure your solution would work under any of these IDEs. Otherwise, I won’t be able to evaluate your solution and you will get zero (0).
* Your source codes should be saved as .cpp or .h file and they are packaged in a ZIP or RAR file. One single ZIP/RAR file should be submitted to the Blackboard electronically.
* It is better if you have source files with proper commenting including your student number, your name, question numbers and explanation for your code.
* Any assignment that is submitted after due date will be punished with 10 points per day. It will be marked as zero after 2 days.
* This is group assignment. Each group should have 2 team members. If you can’t find a partner, you need to let me know in advance.
* You can discuss it with your friends and classmates, but no more than one group should submit the same assignment. If it is done so, students from both groups will be punished with zero.
* Your solution package should include a README.TXT file in the ZIP/RAR file describing any special instructors for me to compile and test your code. README.TXT file should also include name and numbers of group members along with who played what role at what capacity.

Good luck!!!

Adeel Javed

Part Time Instructor - Computer Studies  
Faculty of Continuing Education and Training   
Seneca College