**Programming assignment 2 rubric.**

**Total Points - 100**

| **Criteria** | **Excellent** | **Satisfactory (more points may be awarded depending of number of criteria correctly included)** | **Unsatisfactory** |
| --- | --- | --- | --- |
| Design a **class bankAccount** that defines a bank account as an ADT and implements the basic properties of a bank account.  account holder’s name  account number  balance  interest rate  private static integer to auto-assign account numbers | 11 points –class created all data members included in definition | 5.5 points – missing data members | 0 points – no bankAccount class declared |
| In **bankAccount** class, include a default constructor, the member functions to perform the various operations, required in the menu, on objects of type **bankAccount** (set account data, deposit, withdraw, etc., detailed in part b of the assignment)  Individual get and set functions for all member variables. This will aid in assignment 3 which is based on this assignment | 15 points – properly functioning get and set functions included for all data members; all class data accessible via class functions only and coded correctly | 7.5 points – missing get and set functions for data members; functions not included to perform operations required by menu | 0 points – no get and set function; no class functions included |
| ALL data member variables must be private and accessed through member functions. | 5 points – data members are in the private section | 2.5 points – data member variables declared but are public | 0 points – data member variables not declared |
| Ensure the class declaration and class implementation files are in separate header and code (.cpp) files.  Include the header file in the client program.  Ensure implementation file has class function definitions | 5 points – class declaration is in a header file; class implementation is in an implementation file | 2.5 points – class declaration and implementation in same file separate from client or included in client | 0 points – no class declaration or implementation code written |
| Write a client program that uses the class bankAccount and tests various operations on the objects of the class bankAccount according to the menu in assignment | 5 points – client program included to use the bankAccount class | 1 point – client written but does not use the bankAccount class | 0 points – no client written |
| Declare an array of 20 components of type bankAccount class in the client program (main). Do not create the array inside the class. | 5 points – array of bankAccount of 20 points declared in main | 2.5 points – array declared but not of 20 and/or not in main | 0 points – no array of classes created |
| Menu item: Add a customer. Ensure that the addition of a new customer does not cause an array overflow. The capability of entering the customer name, initial deposit amount and interest should be provided for a new customer. The customer’s account number must be generated from the private static data member and not entered during the ‘add a customer’ action. | 9 points – all requirements met | 4.5 points – array overflow and/or all information entry not provided and/or customer account number not controlled by private static (i.e., entered manually) | 0 points – no requirements met |
| Menu item: Print all customer data for all customers | 5 points – Print all customer data for all customers provided and coded correctly | 2.5 points – Print all customer data for all customers not provided or coded incorrectly | 0 points – no list produced |
| Menu item: Update customer data as follows (submenu):  Display the customers already in the banking system.  Request the user account number to update (you may need to display customer names and numbers for the user to choose) and though a submenu:  (a) Make a deposit for the customer number entered  (b) Make a withdrawal (with error checking) for the customer number entered  (c) Print balance of the customer number entered  (d) Update balance with interest for the customer number entered  (e) Exit the submenu | 2.5 points each step (20)– all provided and coded correctly | 5 points deducted for each requirement not met and/or not correctly coded | 0 points – no requirements met |
| Menu item: Exit program | 5 points – exit provided and coded correctly | 2.5 points – exit not provided or not coded correctly | 0 points – no exit provided |
| Error checking (invalid menu choices, subtraction errors, array limit reached. | 3 points – error checking provided and coded correctly | 1.5 points– error checking not provided or coded incorrectly | 0 points – no error checking |
| README file | 2 points – submission of assignment includes a README file containing assignment-relevant information | 1 points – submission of assignment includes a README file but does not contain relevant information | 0 points – no README file submitted |
| Good programming form | 5 points – program is readable with proper indentation and one line of code per physical line of code | 2.5 points – program has more than one line of code per physical line or does not have proper indentation | 0 points – programming style results in a program that is difficult to read and follow |
| Each function must have required documentation (pre and post conditions). | 5 points – basic comments are included in program as outlined in assignment instructions; all functions have pre/post comments | 2.5 points – comments included but less than minimum requirements; pre/post comments included but not for all function | 0 points – no program comments as requested |