FPP1 — FPP1 TASK 1: CLASS ROSTER

SCRIPTING AND PROGRAMMING APPLICATIONS — C867

PRFA — FPP1

TASK OVERVIEW SUBMISSIONS EVALUATION REPORT

EVALUATION REPORT — ATTEMPT 1 — PASSED

Overall Evaluator Comments

EVALUATOR COMMENTS

C++ skill is evident and the functions are developed to process the data from the array studentData[] with the Williams details. Accessors are developed to handle the instance variables. Tab-formatted output is effective from print(). Validation of the email addresses is included with all three conditions. The subclasses are present to handle the differentiated data for network, software, and security degrees. Additional feedback is provided within the rubric. Good job on the C++ coding for the assessment.

A. PERSONAL INFORMATION

Competent Personal information in the last item of the studentData table is complete.

EVALUATOR COMMENTS: ATTEMPT 1

The last set of data in the table is for Williams, customized as requested in the software design specs.

B. C++ PROJECT

Competent The C++ project is correctly created in the IDE and correctly uses the given files.

There are no comments for this aspect.

C. ENUMERATED DATA TYPES

Competent The enumerated data types for the degree programs are correctly defined.

EVALUATOR COMMENTS: ATTEMPT 1

An enumerated data type is declared for the degree types in the Degree header.

D1. BASE CLASS STUDENT

Competent The base class Student is correctly created by correctly including each of the given variables.

There are no comments for this aspect.

D2A. ACCESSOR

Competent The accessor function in the Student class for each instance variable from part D1 is functional and complete.

EVALUATOR COMMENTS: ATTEMPT 1

Getters are developed for the instance variables in the Student class.

D2B. MUTATOR

Competent The mutator function in the Student class for each instance variable from part D1 is functional or complete.

There are no comments for this aspect.

D2C. CONSTRUCTOR

Competent The constructor function in the Student class accurately uses all of the input parameters from the data table.

There are no comments for this aspect.

D2D. PRINTING SPECIFIC DATA

Competent The virtual print() function in the Student class accurately prints specific student data.

EVALUATOR COMMENTS: ATTEMPT 1

The print() function demonstrates use of accessors to obtain the data for output.

D2E. DESTRUCTOR

Competent The destructor function is implemented to release memory.

There are no comments for this aspect.

D2F. VIRTUAL GETDEGREEPROGRAM

Competent The virtual getDegreeProgram()function is provided with empty implementation.

There are no comments for this aspect.

D3. THREE CLASSES

Competent All 3 classes are a subclass of the Student class using the files created in part B. Each subclass overrides the getDegreeProgram() function. Each subclass has a data member to hold the enumerated type, using the types defined in part C.

There are no comments for this aspect.

E1. ONE ARRAY

Competent The array of pointers created to hold the data in the studentData table is complete and correct.

EVALUATOR COMMENTS: ATTEMPT 1

The Roster class has an array named studentData[] with the sample data.

E2. STUDENT OBJECT

Competent The student object for each student in the data table is correctly created and uses each of the given subclasses. classRosterArray is correctly populated.

There are no comments for this aspect.

E2A. APPLYING POINTER OPERATIONS

Competent The pointer operations are correctly applied when parsing each set of data listed in the studentData table.

There are no comments for this aspect.

E2B. ADDING STUDENT OBJECTS TO ARRAY

Competent Each student object is correctly added to the classRosterArray.

There are no comments for this aspect.

E3A. ADD FUNCTION

Competent The public void add(string studentID, string firstName, string lastName, string emailAddress, int age, int grade1, int grade2, int grade3,) function is correctly defined to set the instance variables from part D1 and update the roster.

There are no comments for this aspect.

E3B. REMOVE FUNCTION

Competent The public void remove(string studentID) that removes students from the roster by student ID is correctly defined.

EVALUATOR COMMENTS: ATTEMPT 1

The data is deleted using studentID as an input parameter in remove() by line 126.

E3C. PRINT ALL FUNCTION

Competent The public void printAll() that prints a complete tab-separated list of student data uses correct accessor functions appropriately. The printAll() correctly loops through all the students in the student list and correctly calls the print() function for each student.

There are no comments for this aspect.

E3D. PRINT AVERAGE FUNCTION

Competent A public void printAverageDaysInCourse(string studentID) correctly prints a student's average number of days in the 3 courses by student ID. The student is correctly identified by the student-ID parameter.

EVALUATOR COMMENTS: ATTEMPT 1

The average days function uses studentID for specific student calculations on line 152.

E3E. PRINT INVALID EMAILS FUNCTION

Competent A public void printInvalidEmails() correctly verifies student email addresses and displays all invalid email addresses to the user.

EVALUATOR COMMENTS: ATTEMPT 1

The function validates email addresses from the array for @, space, and the period domain delimiter.

E3F. PRINT DEGREE PROGRAM FUNCTION

Competent A public void printByDegreeProgram (int degreeProgram) correctly prints out student information for a degree program specified by an enumerated type.

There are no comments for this aspect.

F1. SCREEN PRINT OUT

Competent The course title, programming language used, student ID, and student name are all correctly printed at the top.

EVALUATOR COMMENTS: ATTEMPT 1

The main() function successfully includes the course, student ID, and Williams as a banner.

F2. ROSTER CLASS INSTANCE

Competent The instance of the Roster class called classRoster is correctly created.

EVALUATOR COMMENTS: ATTEMPT 1

An instance of the Roster class called "classRoster" is created in the main() function.

F3. ADD STUDENTS

Competent All students are added to classRoster.

There are no comments for this aspect.

F4. PSEUDO CODE CONVERSION

Competent All pseudo code is correctly converted to complete the rest of the main() function and is in order.

There are no comments for this aspect.

F5. DECONSTRUCTOR CALL

Competent The Roster memory is released by calling the destructor.

EVALUATOR COMMENTS: ATTEMPT 1

A destructor is created for Roster with the prefix tilde and executed to release resources.