Object Oriented Programming – Fall 22

(BS-CS-F22)

Lab-3

Lab Instructor: Maa'm Sanam Ahmad

Instructions:

- **Indent your code properly.**
- **Use meaningful variable and function names.**
- **Use the camelCase notation.**
- **Use meaningful prompt lines/labels for all input/output.**
- **Do NOT use any GLOBAL variable(s). However, global named constants may be used.**
- **This is an individual lab, you are strictly NOT allowed to discuss your solution with fellow colleagues, even not allowed to ask how is he/she is doing, it may result in negative marking. You can ONLY discuss with your TAs or with me. • Anyone caught in an act of plagiarism would be awarded an "F" grade in this Lab.**

Do Validations on inputs where required otherwise 1 mark will be deducted for every wrong validation.

TASK-1:

Task Description:

You are tasked with creating a Student Management System. Define a Student class with the following specifications:

Data Members:

- name: String to store the name of the student.
- cgpa: Double to store the CGPA of the student.
- rollNo: Integer to store the roll number of the student.
- noOfTests: Integer to store the number of tests taken by the student.
- Tests: Integer pointer to store the test scores of the student.
- average: Double to store the average score of the student in all tests.

Constructor:

- Define a default constructor that also behaves as a parametrized constructor with default values for name (an empty string), cgpa (0.0), rollNo (0), noOfTests (0), and the Tests array initialized to nullptr.

Member Functions:

- Setter methods for all the student attributes.
- Getter methods for all data members of the Student (declare getter method as const).
- Destructor that deallocates the Tests array.
- Display function to display information of a Student (name, rollNo, cgpa, and each test score).
- Calculate Average function to calculate the average score for all the tests according to the number of tests.

Driver Main Function:

- Prompt the user to enter the number of students.
- Declare a pointer to Student (Student* students) that points to a dynamically allocated array of students.
- Allocate the array according to the number of students entered by the user.

Global Functions:

- takeInformation(Student* students, int numOfStudents): This function takes a pointer to a Student array and the number of students and asks the user to enter information for every student. After calling this function, the array will be populated with information for every student.
- displayOneStudent(Student* students, int noOfStudents, int studentNo): This function displays information for a particular student number. If the index studentNo is greater than the number of students (studentNo > noOfStudents), the program should display an error message and return from the function.
- displayAllStudents(Student* students, int noOfStudents): This function displays information for all students.
- findTopper(Student* students, int noOfStudents): This function takes the array of all students and the number of students and displays the information of the student who has the highest average score in tests among all students.

Note:

- Make your program menu-based. The takeInformation, displayOneStudent, displayAllStudents, and findTopper functions should be available for execution in the menu.
- If the takeInformation function is called a second time, it should overwrite previous information for all students with new information. The same applies to all other functions.
- Carefully deallocate memory in the destructor as well as the array of students in the main function to avoid dangling pointers.
- (For menu base program use do-while loop).
-

Very information: Carefully deallocate memory in the destructor as well as array of students in main function make pointer nullptr avoiding pointers to become dangling pointers.

In case of a pointer

```
If(ptr!=nullpr)
  {
delete ptr; ptr=nullptr;
}
```

```
// in case of dynamically allocated array
If(ptr!=nullptr)
{
 delete [] ptr;
ptr=nullptr;
}
```

TASK-2:

Task Description:

You are tasked with creating a simple Book Management System. You need to define a Book class with the following specifications:

Data Members:

- title: String to store the title of the book.
- author: String to store the name of the author.
- const int publicationYear: A constant integer to store the publication year of the book.
- float &price: A reference to a floating-point value to store the price of the book.
- int pageCount: Integer to store the number of pages in the book.
- char **contents: Dynamic 2D array to store the contents of the book's pages.

Constructor:

- Define a constructor that initializes all data members of the book. Allocate memory for the contents array based on the number of pages specified.

Accessor Methods:

- getTitle(): Method to retrieve the title of the book.
- getAuthor(): Method to retrieve the author of the book.
- getPublicationYear(): Method to retrieve the publication year of the book.
- getPrice(): Method to retrieve the price of the book.
- getPageCount(): Method to retrieve the number of pages in the book.

Additional Member Functions:

- setPageContent(int page, const string& content): Method to set the content of a specific page.
- displayPageContent(int page): Method to display the content of a specific page.

Conditions to Follow:

- The constructor should allocate memory for the contents array based on the pageCount parameter.
- Ensure proper deallocation of memory in the destructor to avoid memory leaks.

Task Requirements:

- Implement the Book class according to the given specifications, including the constructor, accessor methods, and additional member functions.
- Allocate memory for the contents array in the constructor and deallocate it in the destructor.
- Write a main() function to demonstrate the functionality of the Book class by creating an instance of the class, setting the content of a page, and displaying all relevant information about the book, including the content of a specific page.