**Polytechnic University of Puerto Rico**
**Electrical and Computer Engineering & Computer Science Department**
**CECS 2223 – Computer Programming II Lab**

---

## Lab 3

**Name: Coral S. Schmidt Montilla ID#: 148830**

1. Copy the source code developed for Lab 3 and paste it as **text** below. (*15 points*)

```
/*
* CECS 2223, Computer Programming II Laboratory
* Fall 2023, Sec. 09
* Date: August 30, 2023
* Topic: Lab 3
* File name: Grades.h
* Coral S. Schmidt, #148830
*/
#pragma once
#include <iostream>
using namespace::std;

class Grades {
private:
	static const int TESTS = 3;//Class constant
	int* scores;//Integer array


public:
	Grades();
	//Class Destructor
	~Grades();

	//Class Setters
	void setScores(int);

	//Class Getters
	int getScores(int) const;
	int getAverage() const;
	char getGrades(int) const;

	//Print
	void printGrades(int) const;

};

/*
* CECS 2223, Computer Programming II Laboratory
* Fall 2023, Sec. 09
* Date: August 30, 2023
* Topic: Lab 3
* File name: lab03.cpp
* Coral S. Schmidt Montilla, #148830
*/
#include "Grades.h"
```

---

```
#include <iostream>

/*To implement the class, declare a 5-element Grades array named students. Using a
for iteration control structure, call the appropriate methods to set the scores for
each student and
print the grades.
*/
int main() {

    const int NUM_STUDENTS = 5; // Number of students
    Grades students[NUM_STUDENTS]; // Array of Grades objects

    for (int i = 0; i < NUM_STUDENTS; i++) //gets the scores for each student
    {
        students[i].setScores(i + 1);
        students[i].printGrades(i + 1);
    }

    printf("***** Program developed by Coral S. Schmidt Montilla, ID#148830
*****\n");

    system("pause");
    return 0;
}
/*
* CECS 2223, Computer Programming II Laboratory
* Fall 2023, Sec. 09
* Date: August 30, 2023
* Topic: Lab 3
* File name: Grades.cpp
* Coral S. Schmidt, #148830
*/

#include "Grades.h"

Grades::Grades() {
      scores = new int[TESTS] { 0 };//Initializes the instance variable using
"TESTS" as the size
}

// Class Destructor
Grades::~Grades()//No code needed
{
}

/*The setScores method receives an integer value as parameter and has no return
value.
It prompts the user for a test score, validates the score, and stores it in the
array.The parameter is
the student's number(1, 2, 3, …).It uses a for iteration control structure to prompt
the user
TESTS times using the phrase "Enter the score for student #[student
number], test #[test number]: ". The scores must be in the range [0, 100], and the
```

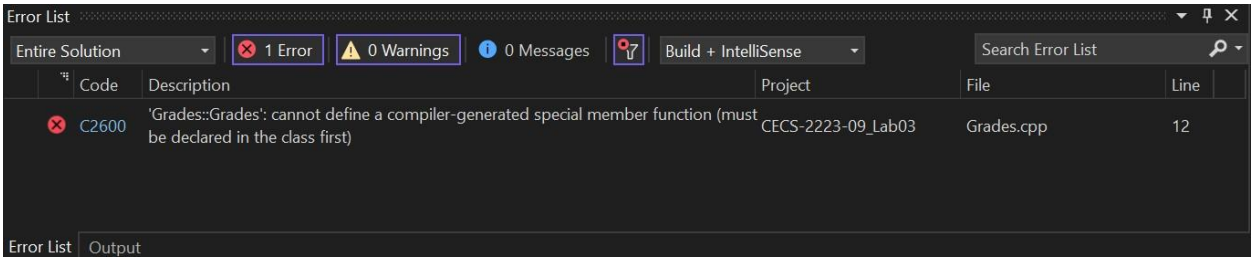
```
method must print an error message indicating that the value is out of range before
requesting a
new value.
*/
void Grades::setScores(int studentNum)
{
    for (int i = 0; i < TESTS; i++) {//prompt the user TESTS times

       //asks user for test score
       printf("Enter the score for student #%i, test #%i:", studentNum, i + 1);
       cin >> scores[i];//saves test score in the array

       if (scores[i] < 0 || scores[i] > 100) //Checks that scores must be in the
range [0, 100] prints error if it isnt in range
            {
             printf("Error: Score is out of range (0-100). Please enter a valid
score.\n");
             i--;
            }
    }
}

//The getScore method receives the index number of the score as parameter and
returns the score.
int Grades::getScores(int indexNum) const
{
    if (indexNum >= 0 && indexNum < TESTS) //Checks if valid index
    {
        return scores[indexNum];
    }
    else // Incase we get an invalid index
    {
        return -1;
    }
}
/*The getAverage method has no parameters and returns an integer value. Using a for
iteration control structure, it adds all test scores and returns the average. Use a
local integer variable
named total to store the sum.*/
int Grades::getAverage() const
{
    int total = 0;

    for (int i = 0; i < TESTS; i++)//Adds all test scores and returns the average
    {
        total += scores[i] / TESTS;
    }

    return total;
}

/*The getGrade method receives an integer value as parameter and returns a character
value. The parameter is the student's number (1, 2, 3, …).It implements the rubric
using an if /
```

```
else if / else selection control structure.*/
char Grades::getGrades(int studentNum) const
{
    if (getAverage() > 89)
    {
        return 'A';
    }
    else if (getAverage() > 79)
    {
        return 'B';
    }
    else if (getAverage() >= 70)
    {
        return 'C';
    }
    else if (getAverage() > 59)
    {
        return 'D';
    }
    else
    {
        return 'F';
    }
}

/*The printGrades method receives an integer value as parameter and has no return
value. The parameter is the student's number (1, 2, 3, …). It prints the phrase
"Student
#[student number], with test scores [score1], [score2], and
[score3], scored a final average of [final average] and earned
a(n) [letter grade]." Make sure to add a blank line before and after the phrase is
printed.*/
void Grades::printGrades(int studentNum) const
{
    printf("\nStudent #%i, with test scores: ", studentNum);

    for (int i = 0; i < TESTS; i++)
    {
        printf("%i", scores[i]);

        if (i < TESTS - 1)
        {
            printf(", ");
        }
    }
    printf(", scored a final average of %i and earned a(n) %c.\n", getAverage(),
getGrades(studentNum));
}
```

2. Paste the screenshots of the program's execution below. (*5 points*)

```
C:\Users\coral\Desktop\Politecnica\Computer Science\Computer Programing Lab II\CECS-2223-09_Lab03\x64\Debug\CECS-2223-09_Lab03.exe
Enter the score for student #1, test #1:-8
Error: Score is out of range (0-100). Please enter a valid score.
Enter the score for student #1, test #1:800
Error: Score is out of range (0-100). Please enter a valid score.
Enter the score for student #1, test #1:7600
Error: Score is out of range (0-100). Please enter a valid score.
Enter the score for student #1, test #1:100
Enter the score for student #1, test #2:99
Enter the score for student #1, test #3:92

Student #1, with test scores: 100, 99, 92, scored a final average of 96 and earned a(n) A.
Enter the score for student #2, test #1:88
Enter the score for student #2, test #2:81
Enter the score for student #2, test #3:79

Student #2, with test scores: 88, 81, 79, scored a final average of 82 and earned a(n) B.
Enter the score for student #3, test #1:79
Enter the score for student #3, test #2:80
Enter the score for student #3, test #3:68

Student #3, with test scores: 79, 80, 68, scored a final average of 74 and earned a(n) C.
Enter the score for student #4, test #1:62
Enter the score for student #4, test #2:100
Enter the score for student #4, test #3:0

Student #4, with test scores: 62, 100, 0, scored a final average of 53 and earned a(n) F.
Enter the score for student #5, test #1:70
Enter the score for student #5, test #2:100
Enter the score for student #5, test #3:40

Student #5, with test scores: 70, 100, 40, scored a final average of 69 and earned a(n) D.
***** Program developed by Coral S. Schmidt Montilla, ID#148830 *****
Press any key to continue . . .
```

3. Comment on any warnings or errors revealed by Visual Studio. If any error messages were present, list the error and describe how you corrected it. If no errors or warnings were revealed, comment on the most important aspect of developing the solution. (*5 points*)




Code with error:

```
public:
	//Class Destructor
	~Grades();

	//Class Setters
```

```
        void setScores(int);

        //Class Getters
        int getScores(int) const;
        int getAverage() const;
        char getGrades(int) const;

        //Print
        void printGrades(int) const;

};
```

Fixed:

```
public:
        Grades();
        //Class Destructor
        ~Grades();

        //Class Setters
        void setScores(int);

        //Class Getters
        int getScores(int) const;
        int getAverage() const;
        char getGrades(int) const;

        //Print
        void printGrades(int) const;

        };
```

I added Grades(); to Grades.h to fix the error.