# Assignment Description

Write a modular program that analyzes a student’s quiz scores (0-15). In addition to main, the program should have a getData function that accepts the quiz scores for 16 weeks from the user and stores it in a template array. It should also have two value-returning functions that compute and return to main:

* highScore, and
* lowScore,

These functions return the number of the module with the lowest and highest quiz scores, not the quiz scores for those modules. Notice that this module number can be used to obtain the quiz score for that module. This information should be used by your main program to print a summary quiz report similar to the following:

Fall 2019 Quiz Grades:

            Module 1:   9

            Module 2:  15

            …

            Module 16: 12

Average Grade: 13

Your lowest score was in Module 5 with 0.

Your highest score was in Module 2 with 15.

# 1 Readme Documentation

This program will get input for 16 grades, and then calculate the high, low, and average score.

# 2 Flowchart Screen Shots

# 3 UML and Use Case Diagrams

# 4 Source Code of All files (.h, .cpp)

#include *<iostream>*

#include *<iomanip>*

#include *<string>*

#include *<cctype>*

#include *<stdexcept>*

#include *<array>*

#include *<vector>*

*/\**

*Name: Grades Statistics*

*Author: Wesley Hixon*

*Date Last Updated: 11/08/2024*

*Purpose: This program will take input for 16 quiz grades.*

*Then it will output the highest score, the lowest score, and the average score.*

*\*/*

**using** **namespace** **std**;

*// Function definitions*

void getData(array<int, 16>& grades);

int highScore(**const** array<int, 16>& grades);

int lowScore(**const** array<int, 16>& grades);

int averageScore(**const** array<int, 16>& grades);

int main(){

*// Creating grades array and getting data*

array<int, 16> grades;

getData(grades);

*// Finding max, min, and average score*

int maxIndex = highScore(grades);

int minIndex = lowScore(grades);

int average = averageScore(grades);

*// Outputting all grades*

cout << endl << "Quiz Grades Report:" << endl;

**for**(int i = 0; i < 16; i++){

cout << "**\t**Module " << i + 1 << ": " << grades[i] << endl;

}

*// Outputting average, low, and high scores*

cout << "Average grade: " << average << endl;

cout << "Your lowest score was in Module " << minIndex + 1 << " with a " << grades[minIndex] << "." << endl;

cout << "Your highest score was in Module " << maxIndex + 1 << " with a " << grades[maxIndex] << "." << endl;

**return** 0;

}

*// Accept quiz scores for 16 weeks and stores in template array*

void getData(array<int, 16>& grades){

int moduleIndex = 0;

bool valid = false;

*// Iterating through each grade in array using for loop index*

**for**(moduleIndex; moduleIndex < 16; moduleIndex++){

cout << "Please enter the quiz grade for Module " << moduleIndex + 1 << ": " << endl;

*// Getting valid grade greater than 0 and storing it in array*

valid = false;

**while**(!valid){

**try**{

cin >> grades[moduleIndex];

**if**(cin.fail() || grades[moduleIndex] < 0){

cin.clear();

cin.ignore(10000, '\n');

**throw**(runtime\_error("Please enter a valid grade of at least 0."));

}

valid = true;

}**catch**(**const** exception& e){

cerr << endl << e.what() << endl;

}

}

}

**return**;

}

*// Returns the index of the highest score in grades array*

int highScore(**const** array<int, 16>& grades){

int max = 0;

int maxIndex = 0;

**for**(int i = 0; i < 16; i++){

**if**(grades[i] > max){

max = grades[i];

maxIndex = i;

}

}

**return** maxIndex;

}

*// Returns the index of the lowest score in grades array*

int lowScore(**const** array<int, 16>& grades){

int min = grades[0];

int minIndex = 0;

**for**(int i = 0; i < 16; i++){

**if**(grades[i] < min){

min = grades[i];

minIndex = i;

}

}

**return** minIndex;

}

*// Finds the average score of all grades*

int averageScore(**const** array<int, 16>& grades){

int average;

int sum = 0;

*// Finding sum*

**for**(int i = 0; i < 16; i++){

sum += grades[i];

}

*// Finding average*

average = sum/16;

**return** average;

}

# 5 Three Use Case Screen Shots







