Humna Sultan
Mrs. DiGiovanna
CompSci 10B
6 November 2019

Honor Roll: Coder/Tester Program

Green Color - Procedural
Red Color - Object Oriented

Procedural

honor_roll

Source Code:

```
/*Name: Humna Sultan
 *6 November 2019
 *Honor Roll Coder/Tester 1 - PROCEDURAL
Requirements:
 1. Student must have a 90 average, at least 5 classes, and no
infractions for eligibility for honor roll
 2. Maximum of 8 courses/Minimum of 1 Course
 3. Input includes name, classes, and grades
 4. Class name is limited to 20 characters
 5. Grades are integers - Maximum 100/Minimum 0
 6. Average is rounded to the nearest integer
 7. Discipline issue is randomly assigned
 8. Random assignment is truly random and properly coded
 9. Inappropriate values entered results in the user being able
to re-input info
 10. Values that are not appropriate to a certain variable type
produce no errors - user is able to re-enter info
 11. All infomation is printed after input is taken
 12. Disciplinary status and Average are printed
 13. Output is neat, organized, formatted correctly
 14. All outputs are included
 15. Procedural and Object Oriented are both included and
submitted
 16. Comments are descriptive
 17. Header includes name, date, lab name, requirements
```

```
 18. Invalid input messages are included if input is not
appropriate
*/

#include <iostream>
#include <string>
#include <sstream>
#include <algorithm>
#include <cstdlib>
#include <cmath>
#include <stdio.h>
#include <time.h>
#include "GetData.h"
//includes

using namespace std;
//namespace

int main()
//main
{
     string name;
     cout << "Enter your name :: ";
     getline(cin, name);
     //asks for name and stores

     bool isValid = false;
     //boolean - used for GetData
     int cour_num;
     //variable for number of courses
     do
     {
          do
          {
               cout << "How many courses are you currently
taking? (MAX 8; MIN 1) :: ";
               //asks for number of courses
               isValid = getValidInt(cour_num);
```

```
               //sets isValid to boolean return statement for
GetData (checks if integer)
               if (isValid == false)
               {
                    cout << "You entered a number of classes
that is not a positive integer! Please try again!" << endl;
               }
               //invalid input message
          } while (isValid == false);
          //do while for appropriate integer response
          if (cour_num > 8)
          {
               cout << "You entered a number of classes that is
greater than 8! Please try again!" << endl;
          }
          //invalid input message
          if (cour_num < 1)
          {
               cout << "You entered a number of classes that is
less than 1! Please try again!" << endl;
          }
          //invalid input message
     } while (cour_num > 8 || cour_num < 1);
     //do while for integer that is between appropriate values



     string courses[8];
     //string array for course names
     int counter = 1;
     //sets counter to 1
     for (int a = 0; a < cour_num; a++)
          //for loop; goes for number of courses
     {
          do
          {
               cout << "Enter the name of course number " <<
counter << " (limit 20 characters) :: ";
               getline(cin, courses[a]);
```

```
                //asks for name of course and stores in array
                counter++;
                //increments counter
                if (courses[a].length() > 20)
                {
                     counter--;
                     cout << "The course name that you entered is
more than 20 characters! Please try again!" << endl;
                }
                //invalid input message
                //if course name > 20 --> decrements counter back
to what it was before
          } while (courses[a].length() > 20);
          //do while for max course name length being 20
     }

     int grades[8];
     //array for storing grades
     string grade = "";
     //grade string for input
     bool isValid2 = false;
     //boolean - used for GetData
     for (int i = 0; i < cour_num; i++)
          //for loop; goes until number of courses
     {
          do
          {
                do {
                     cout << "Enter the grade you receieved for "
<< courses[i] << " (integer values 0-100) :: ";
                     //asks for grade
                     isValid2 = getValidInt(grades[i]);
                     //sets isValid2 to boolean return statement
for GetData (checks if integer)
                     if (isValid2 == false)
                     {
                          cout << "You entered a grade that is
not a positive integer value! Please try again!" << endl;
                     }
```

```
                    //invalid input message
               } while (isValid2 == false);
               //do while for appropriate integer response
               if (grades[i] > 100)
               {
                    cout << "You entered a grade that is greater
than 100! Please try again!" << endl;
               }
               //invalid input message
          } while (grades[i] < 0 || grades[i] > 100);
          //do while for integer that is between appropriate
values
     }
     srand(time(NULL));
     //random
     int random = rand() % 2;
     //sets random to random num %2 - 0 or 1
     string disc;
     //string disc used for yes or no discipline infraction
     if (random == 0)
     {
          disc = "NO";
     }
     //if random == 0 - no infraction
     if (random == 1)
     {
          disc = "YES";
     }
     //if random == 1 - infraction

     int total = 0;
     //total - sum of all grades
     for (int j = 0; j < cour_num; j++)
          //for loop that goes until number of courses
     {
          total += grades[j];
          //adds the value for each grade to total
     }
     int average = total / cour_num;
```

```
     //calulates average

     cout << "\n\nName:" << name << "\n" << endl;
     //prints name
     cout << "Class                    Grade" << endl;
     //prints class and grade with spaces
     int count = 1;
     //sets count
     int spaces;
     //sets spaces
     for (int k = 0; k < cour_num; k++)
     {
          //for loop for spacing
          spaces = 25 - courses[k].length();
          //calculates number of spaces needed between course
name and grade
          cout << courses[k];
          for (int e = 0; e < spaces; e++) {
               cout << " ";
          }
          //prints spaces
          cout << grades[k] << endl;
          //prints grade value
     }
     cout << "\nAverage: " << average << endl;
     cout << "Disciplinary Infraction: " << disc << endl;
     //prints average and infraction status

     if (cour_num >= 5 && average >= 90 && random == 0)
     {
          cout << "Congratulations, " << name << "! You have
made the honor roll.";
     }
     //if user meets requirements for honor roll
     if (cour_num < 5 || average < 90 || random == 1)
     {
          cout << "I'm sorry, " << name << ", but you didn't
qualify for the honor roll.";
     }
```

```
     //if user does not meet requirements for honor roll

     return 0;
     //return statement
}
```

GetData.h

Source Code:

```
/*Name: Humna Sultan
 *6 November 2019
 *Honor Roll Coder/Tester 1 - PROCEDURAL
Requirements:
 1. Student must have a 90 average, at least 5 classes, and no
infractions for eligibility for honor roll
 2. Maximum of 8 courses/Minimum of 1 Course
 3. Input includes name, classes, and grades
 4. Class name is limited to 20 characters
 5. Grades are integers - Maximum 100/Minimum 0
 6. Average is rounded to the nearest integer
 7. Discipline issue is randomly assigned
 8. Random assignment is truly random and properly coded
 9. Inappropriate values entered results in the user being able
to reinput info
 10. Values that are not appropriate to a certain variable type
produce no errors - user is able to re-enter info
 11. All infomation is printed after input is taken
 12. Disciplinary status and Average are printed
 13. Output is neat, organized, formatted correctly
 14. All outputs are included
 15. Procedural and Object Oriented are both included and
submitted
 16. Comments are descriptive
 17. Header includes name, date, lab name, requirements
 18. Invalid input messages are included if input is not
appropriate
*/
```

```
#pragma once
#include <iostream>
//iostream
using namespace std;
//namespace

bool getValidInt(int&);
bool getValidFloat(float&);
bool getValidDouble(double&);
//methods in GetData
```

GetData.cpp

Source Code:

```
/*Name: Humna Sultan
 *6 November 2019
 *Honor Roll Coder/Tester 1 - PROCEDURAL
Requirements:
 1. Student must have a 90 average, at least 5 classes, and no
infractions for eligibility for honor roll
 2. Maximum of 8 courses/Minimum of 1 Course
 3. Input includes name, classes, and grades
 4. Class name is limited to 20 characters
 5. Grades are integers - Maximum 100/Minimum 0
 6. Average is rounded to the nearest integer
 7. Discipline issue is randomly assigned
 8. Random assignment is truly random and properly coded
 9. Inappropriate values entered results in the user being able
to reinput info
 10. Values that are not appropriate to a certain variable type
produce no errors - user is able to re-enter info
 11. All infomation is printed after input is taken
 12. Disciplinary status and Average are printed
 13. Output is neat, organized, formatted correctly
 14. All outputs are included
 15. Procedural and Object Oriented are both included and
submitted
 16. Comments are descriptive
```

```
 17. Header includes name, date, lab name, requirements
 18. Invalid input messages are included if input is not
appropriate
*/

#include "GetData.h"
#include <string>
//includes
using namespace std;
//namespace

bool getValidInt(int& t_int) {
     //used for getting valid integers
     string intScratch = "";
     bool isValid = false;

     getline(cin, intScratch);
     isValid = intScratch.find_first_not_of("0123456789") ==
string::npos;
     if (isValid) {
          t_int = stoi(intScratch);
     }
     //uses stoi in order to check if value is appropriate
     return isValid;
     //returns boolean isValid true/false
}
bool getValidFloat(float& t_float) {
     //used for getting valid floats
     string floatScratch = "";
     bool isValid = false;

     getline(cin, floatScratch);
     isValid = floatScratch.find_first_not_of("0123456789.") ==
string::npos;
     if (isValid) {
          t_float = stof(floatScratch);
     }
     //uses stof in order to check if value is appropriate
     return isValid;
```

```
     //returns boolean isValid true/false
}
bool getValidDouble(double& t_double) {
     //used for getting valid doubles
     string doubleScratch = "";
     bool isValid = false;

     getline(cin, doubleScratch);
     isValid = doubleScratch.find_first_not_of("0123456789.") ==
string::npos;
     if (isValid) {
          t_double = stod(doubleScratch);
     }
     //uses stod in order to check if value is appropriate
     return isValid;
     //returns boolean isValid true/false
}
```

Object Oriented

honor_roll2.cpp:

Source Code:

```
/*Name: Humna Sultan
 *6 November 2019
 *Honor Roll Coder/Tester 1 - OBJECT ORIENTED
Requirements:
 1. Student must have a 90 average, at least 5 classes, and no
infractions for eligibility for honor roll
 2. Maximum of 8 courses/Minimum of 1 Course
 3. Input includes name, classes, and grades
 4. Class name is limited to 20 characters
 5. Grades are integers - Maximum 100/Minimum 0
 6. Average is rounded to the nearest integer
 7. Discipline issue is randomly assigned
 8. Random assignment is truly random and properly coded
```

```
 9. Inappropriate values entered results in the user being able
to re-input info
 10. Values that are not appropriate to a certain variable type
produce no errors - user is able to re-enter info
 11. All infomation is printed after input is taken
 12. Disciplinary status and Average are printed
 13. Output is neat, organized, formatted correctly
 14. All outputs are included
 15. Procedural and Object Oriented are both included and
submitted
 16. Comments are descriptive
 17. Header includes name, date, lab name, requirements
 18. Invalid input messages are included if input is not
appropriate
*/

#include <iostream>
#include <string>
#include <sstream>
#include <algorithm>
#include <cstdlib>
#include <cmath>
#include <stdio.h>
#include <time.h>
#include "GetData2.h"
#include "honor_roll_class.h"
//includes

using namespace std;
//namespace

int main()
//main
{
     honor_roll_class h = honor_roll_class();
     //creates object of honor roll class

     h.name();
     //calls name function
```

```
     return 0;
     //return statement
}
```

honor_roll_class.cpp:

Source Code:

```
/*Name: Humna Sultan
 *6 November 2019
 *Honor Roll Coder/Tester 1 - OBJECT ORIENTED
Requirements:
 1. Student must have a 90 average, at least 5 classes, and no
infractions for eligibility for honor roll
 2. Maximum of 8 courses/Minimum of 1 Course
 3. Input includes name, classes, and grades
 4. Class name is limited to 20 characters
 5. Grades are integers - Maximum 100/Minimum 0
 6. Average is rounded to the nearest integer
 7. Discipline issue is randomly assigned
 8. Random assignment is truly random and properly coded
 9. Inappropriate values entered results in the user being able
to re-input info
 10. Values that are not appropriate to a certain variable type
produce no errors - user is able to re-enter info
 11. All infomation is printed after input is taken
 12. Disciplinary status and Average are printed
 13. Output is neat, organized, formatted correctly
 14. All outputs are included
 15. Procedural and Object Oriented are both included and
submitted
 16. Comments are descriptive
 17. Header includes name, date, lab name, requirements
 18. Invalid input messages are included if input is not
appropriate
*/

#include "honor_roll_class.h"
```

```
#include "GetData2.h"
#include <iomanip>
#include <string>
//includes

using namespace std;
//namespace

void honor_roll_class::name() {
     //method for getting user's name

     string name;
     cout << "Enter your name:: ";
     getline(cin, name);
     //asks for name and stores in variable

     coursenum(name);
     //proceeds to method for finding the number of courses
}

void honor_roll_class::coursenum(string name)
//method for finding the number of courses
{
     bool isValid = false;
     //boolean - used for GetData2
     int cour_num;
     //variable for number of courses
     do
     {
          do
          {
               cout << "How many courses are you currently
taking? (MAX 8; MIN 1) :: ";
               //asks for number of courses
               isValid = getValidInt(cour_num);
               //sets isValid to boolean return statement for
GetData (checks if integer)
               if (isValid == false)
               {
```

```
                    cout << "You entered a number of classes
that is not a positive integer! Please try again!" << endl;
               }
          } while (isValid == false);
          //do while for appropriate integer response
          if (cour_num > 8)
          {
               cout << "You entered a number of classes that is
greater than 8! Please try again!" << endl;
          }
          //invalid input message
          if (cour_num < 1)
          {
               cout << "You entered a number of classes that is
less than 1! Please try again!" << endl;
          }
          //invalid input message
     } while (cour_num > 8 || cour_num < 1);
     //do while for integer that is between appropriate values

     courses(cour_num, name);
     //proceeds to method for finding names of courses
}

void honor_roll_class::courses(int cour_num, string name)
//method for finding course names
{
     string courses[8];
     //string array for course names
     int counter = 1;
     //sets counter to 1
     for (int a = 0; a < cour_num; a++)
          //for loop; goes for number of courses
     {
          do
          {
               cout << "Enter the name of course number " <<
counter << " (limit 20 characters) :: ";
               getline(cin, courses[a]);
```

```
                //asks for name of course and stores in array
                counter++;
                //increments counter
                if (courses[a].length() > 20)
                {
                     counter--;
                     cout << "The course name that you entered is
more than 20 characters! Please try again!" << endl;
                }
                //invalid input message
                //if course name > 20 --> decrements counter back
to what it was before
          } while (courses[a].length() > 20);
          //do while for max course name length being 20
     }
     grades(cour_num, courses, name);
     //proceeds to method for finding grades of courses
}

void honor_roll_class::grades(int cour_num, string courses[],
string name)
//method for getting grades for each course
{
     bool isValid2 = false;
     //boolean - used for GetData2
     int grades[8];
     //array for storing grades
     string grade = "";
     //grade string for input
     for (int i = 0; i < cour_num; i++)
          //for loop; goes until number of courses
     {
          do
          {
                do {
                     cout << "Enter the grade you receieved for "
<< courses[i] << " (integer values 0-100):: ";
                     //asks for grade
                     isValid2 = getValidInt(grades[i]);
```

```
                    //sets isValid2 to boolean return statement
for GetData2 (checks if integer)
                    if (isValid2 == false)
                    {
                         cout << "You entered a grade that is
not a positive integer value! Please try again!" << endl;
                    }
                    //invalid input message
               } while (isValid2 == false);
               //do while for appropriate integer response
               if (grades[i] > 100)
               {
                    cout << "You entered a grade that is greater
than 100! Please try again!" << endl;
               }

          } while (grades[i] < 0 || grades[i] > 100);
          //do while for integer that is between appropriate
values
     }
     random(cour_num, courses, grades, name);
     //proceeds for method for random generation
}

void honor_roll_class::random(int cour_num, string courses[],
int grades[], string name)
//method for assigning disciplinary infraction
{
     srand(time(NULL));
     //random
     int random = rand() % 2;
     //sets random to random num %2 - 0 or 1
     string disc;
     //string disc used for yes or no discipline infraction
     if (random == 0)
     {
          disc = "NO";
     }
     //if random == 0 - no infraction
```

```
	if (random == 1)
	{
		disc = "YES";
	}
	//if random == 1 - infraction

	printInfo(name, cour_num, grades, courses, disc, random);
	//proceeds to method for printing information
}

void honor_roll_class::printInfo(string name, int cour_num, int
grades[], string courses[], string disc, int random)
//method for printing information
{
	double total = 0;
	//total - sum of all grades
	for (int j = 0; j < cour_num; j++)
		//for loop that goes until the number of courses
	{
		total += grades[j];
		//adds the value of each grade to total
	}
	int average = total / cour_num;
	//calculates average

	cout << "\n\nName:" << name << "\n" << endl;
	//prints name
	cout << "Class                    Grade" << endl;
	//prints class and grade with spaces
	int count = 1;
	//sets count
	int spaces;
	//sets spaces
	for (int k = 0; k < cour_num; k++)
	{
		//for loop for spacing
		spaces = 25 - courses[k].length();
		//calculates number of spaces needed between course
name and grade
```

```
            cout << courses[k];
            //prints course name
            for (int e = 0; e < spaces; e++) {
                 cout << " ";
            }
            //prints spaces
            cout << grades[k] << endl;
            //prints grade value
      }
      cout << "\nAverage: " << average << endl;
      cout << "Disciplinary Infraction: " << disc << endl;
      //prints average and infraction status

      if (cour_num >= 5 && average >= 90 && random == 0)
      {
            cout << "Congratulations, " << name << "! You have
made the honor roll.";
      }
      //if user meets requirements for honor roll
      if (cour_num < 5 || average < 90 || random == 1)
      {
            cout << "I'm sorry, " << name << ", but you didn't
qualify for the honor roll.";
      }
      //if user does not meet requirements for honor roll
}
```

GetData2.cpp:

Source Code:

```
/*Name: Humna Sultan
 *6 November 2019
 *Honor Roll Coder/Tester 1 - OBJECT ORIENTED
Requirements:
 1. Student must have a 90 average, at least 5 classes, and no
infractions for eligibility for honor roll
 2. Maximum of 8 courses/Minimum of 1 Course
 3. Input includes name, classes, and grades
```

```
 4. Class name is limited to 20 characters
 5. Grades are integers - Maximum 100/Minimum 0
 6. Average is rounded to the nearest integer
 7. Discipline issue is randomly assigned
 8. Random assignment is truly random and properly coded
 9. Inappropriate values entered results in the user being able
to re-input info
 10. Values that are not appropriate to a certain variable type
produce no errors - user is able to re-enter info
 11. All infomation is printed after input is taken
 12. Disciplinary status and Average are printed
 13. Output is neat, organized, formatted correctly
 14. All outputs are included
 15. Procedural and Object Oriented are both included and
submitted
 16. Comments are descriptive
 17. Header includes name, date, lab name, requirements
 18. Invalid input messages are included if input is not
appropriate
*/

#include "GetData2.h"
#include <string>
//includes

using namespace std;
//namespace

bool getValidInt(int& t_int) {
     //used for getting valid integers
     string intScratch = "";
     bool isValid = false;

     getline(cin, intScratch);
     isValid = intScratch.find_first_not_of("0123456789") ==
string::npos;
     if (isValid) {
          t_int = stoi(intScratch);
     }
```

```
	//uses stoi in order to check if value is appropriate

	return isValid;
	//returns boolean isValid true/false
}
bool getValidFloat(float& t_float) {
	//used for getting valid floats
	string floatScratch = "";
	bool isValid = false;

	getline(cin, floatScratch);
	isValid = floatScratch.find_first_not_of("0123456789.") ==
string::npos;
	if (isValid) {
		t_float = stof(floatScratch);
	}
	//uses stof in order to check if value is appropriate

	return isValid;
	//returns boolean isValid true/false
}
bool getValidDouble(double& t_double) {
	//used for getting valid doubles
	string doubleScratch = "";
	bool isValid = false;

	getline(cin, doubleScratch);
	isValid = doubleScratch.find_first_not_of("0123456789.") ==
string::npos;
	if (isValid) {
		t_double = stod(doubleScratch);
	}
	//uses stod in order to check if value is appropriate

	return isValid;
	//returns boolean isValid true/false
}
```

GetData2.h:

Source Code:

```
/*Name: Humna Sultan
 *6 November 2019
 *Honor Roll Coder/Tester 1 - OBJECT ORIENTED
Requirements:
 1. Student must have a 90 average, at least 5 classes, and no
infractions for eligibility for honor roll
 2. Maximum of 8 courses/Minimum of 1 Course
 3. Input includes name, classes, and grades
 4. Class name is limited to 20 characters
 5. Grades are integers - Maximum 100/Minimum 0
 6. Average is rounded to the nearest integer
 7. Discipline issue is randomly assigned
 8. Random assignment is truly random and properly coded
 9. Inappropriate values entered results in the user being able
to re-input info
 10. Values that are not appropriate to a certain variable type
produce no errors - user is able to re-enter info
 11. All infomation is printed after input is taken
 12. Disciplinary status and Average are printed
 13. Output is neat, organized, formatted correctly
 14. All outputs are included
 15. Procedural and Object Oriented are both included and
submitted
 16. Comments are descriptive
 17. Header includes name, date, lab name, requirements
 18. Invalid input messages are included if input is not
appropriate
*/

#pragma once
#include <iostream>
//iostream
using namespace std;
//namespace

bool getValidInt(int&);
bool getValidFloat(float&);
```

```
bool getValidDouble(double&);
//methods in GetData2
```

honor_roll_class.h:

Source Code:

```
/*Name: Humna Sultan
 *6 November 2019
 *Honor Roll Coder/Tester 1 - OBJECT ORIENTED
Requirements:
 1. Student must have a 90 average, at least 5 classes, and no
infractions for eligibility for honor roll
 2. Maximum of 8 courses/Minimum of 1 Course
 3. Input includes name, classes, and grades
 4. Class name is limited to 20 characters
 5. Grades are integers - Maximum 100/Minimum 0
 6. Average is rounded to the nearest integer
 7. Discipline issue is randomly assigned
 8. Random assignment is truly random and properly coded
 9. Inappropriate values entered results in the user being able
to re-input info
 10. Values that are not appropriate to a certain variable type
produce no errors - user is able to re-enter info
 11. All infomation is printed after input is taken
 12. Disciplinary status and Average are printed
 13. Output is neat, organized, formatted correctly
 14. All outputs are included
 15. Procedural and Object Oriented are both included and
submitted
 16. Comments are descriptive
 17. Header includes name, date, lab name, requirements
 18. Invalid input messages are included if input is not
appropriate
*/

#pragma once
#include <iostream>
//includes
```

```
using namespace std;
//namespace

class honor_roll_class
     //class name
{
public:
     //public functions
     void name();
     void coursenum(string);
     void courses(int, string);
     void grades(int, string[], string);
     void random(int, string[], int[], string);
     void printInfo(string, int, int[], string[], string, int);
     //declares functions for cpp honor_roll_class
};
```