VNU – HCMC UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

PROJECT REPORT 1:

MOCK STUDENT DATA GENERATOR

Instructor guide: Tran Duy Quang

Students:

- **Nguyen Nhat Truong – 20120229**
- **Ngo Nguyen Quang Tu – 20120234**

Class: OOP 20_3

TABLE OF CONTENT:

MOCK STUDENT DATA GENERATOR

I/ GROUP INFORMATION:

1. Students infomation:

Full name	ID
Nguyen Nhat Truong	20120229
Ngo Nguyen Quang Tu	20120234

II/ ASSESSMENT OF COMPLIANCES:

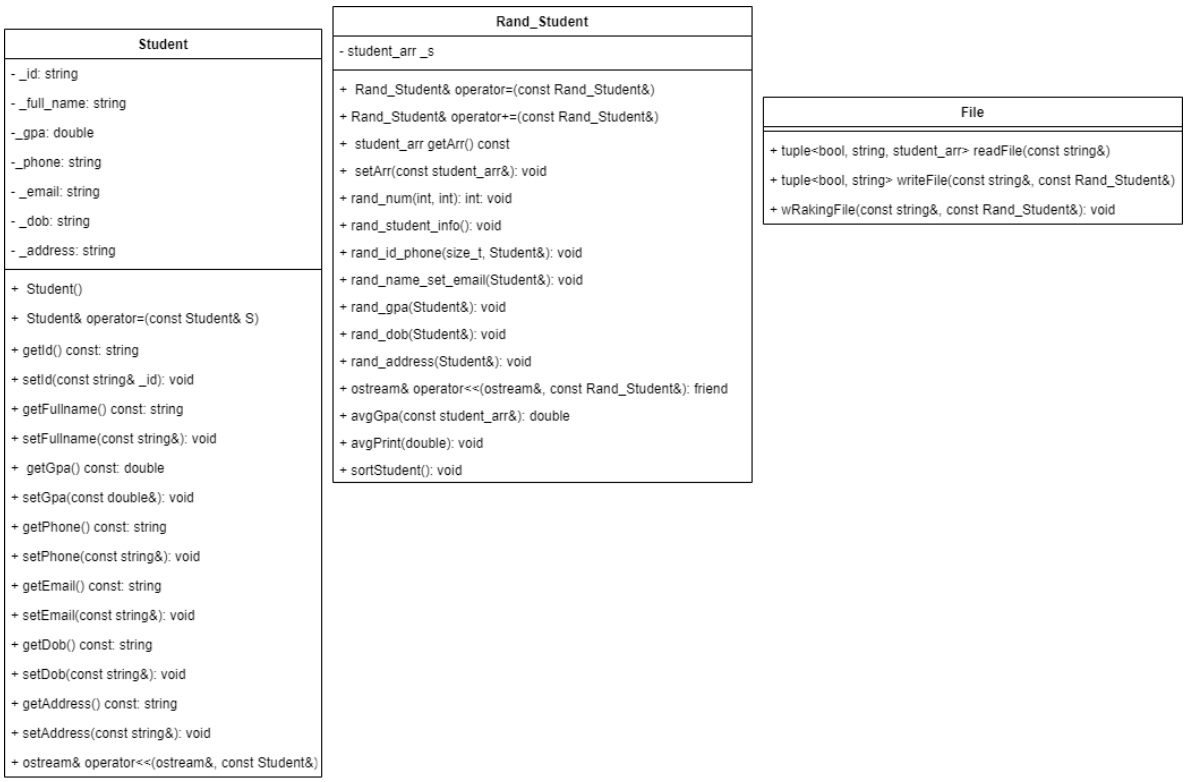
Complete 100% of the requirements of the project.

III/ PROJECT:

1. Main Target:

- Create Simple Classes (File.h, Rand_Student.h, Student.h ,..)
- Practice On Handling Data Stream By tuple<classs T1,class T2,..> (when reading & writing file **.txt**)
- Using rand() Function To Generate Data Randomly (for every each student's infor)
- Read A File In JSON Format (for address random generator)
- Validate Check: Indicate The **Error Line** Position In Text If Wrongly Formatted (bonus function)
- Sort A Vector And Save As A Ranked List In **.csv** Format (bonus function)

2. Class diagram:



3. Working Progress:

After project, we both appreciate our experience on our team-working time which we are so grateful for and proud of ourselves. We are on 200% of our ability way to have accomplished one as soon as possible. Hence, all requirements described in Project Instruction are completely met.

Basic Requirements	Done Status
Read all students saved in the file "students.txt" back into a vector of Student	Yes
Generate randomly a number n in the range of [5, 10]	Yes
Generate randomly n Students and add to the previous vector	Yes
Overwrite and save and the students in the current vector back to the file "students.txt"	Yes
Print out the average GPA of all students	Yes
Print out all the students that have a GPA greater than the average GPA	Yes

Bonus Functions

Not only building the essential-coded function, we also add an external lib to help make a better random for addresses that are written in JSON format as well. In addition, there are some bonus functions to check the wrongly-formatted line in file **students.txt** (validate GPA, telephone, email, dob) then indicate where that line is through showing a message on console with line index. If prgramm runs without no failure, based on their own GPA a *.csv* ranked list will be released demonstrating the TOP students sorted ASC with ranked order.

Sample Ranked List of T.O.P GPA Students:

rankings

ID	FULL NAME	GPA	RANKED
20120583	Luong Thoi Phuoc	9.990000	#1
20120608	Bui Huu Minh	9.520000	#2
2127832	Nguyen Dieu Linh	9.220000	#3
20120993	Luong Ngo Phuoc	8.360000	#4
2127833	Ngo Thoi Chanh	8.120000	#5
2127831	Tran Huu Minh	7.300000	#6
20120629	Duong Quang Minh	5.520000	#7
20120077	Bui Cam Linh	4.470000	#8

4. Project construction steps:

a/ In the header files:

- **Student.h:**

  + Call libraries and declare setter, getter, constructor, and destructor functions in the public part of the class:

```
public: //constructor,destructor
    Student();
    Student(const string&, const string&, const double&, const string&, const string&, const string&, const string&);
    Student(const Student& S);
    ~Student() {}
public: //setter / getter
    Student& operator=(const Student& S); //opeator = defined function for class

    string getId() const;
    void setId(const string& _id);

    string getFullname() const;
    void setFullname(const string&);

    double getGpa() const;
    void setGpa(const double&);

    string getPhone() const;
    void setPhone(const string&);

    string getEmail() const;
    void setEmail(const string&);

    string getDob() const;
    void setDob(const string&);

    string getAddress() const;
    void setAddress(const string&);
```

  + Use ostream and operator to print out all students' information:

```
friend ostream& operator<<(ostream&, const Student&);
```

  + Declare class properties in private consists of identity string, full name, gpa must be from [0.0,10.0], phone be a range of 10 numbers which has begining num range of "09", email of student based on full name, end up with "@student.hcmus.edu.vn", birthday and address of students at TPHCM city:

```
private:
    string _id; //identity string
    string _full_name; //full name
    double _gpa; //gpa must be from [0.0,10.0]
    string _phone; //phone be a range of 10 numbers which has begining num range of "09"
    string _email; //email of student based on full name, end up with "@student.hcmus.edu.vn"
    string _dob; //birthday
    string _address; //address of students at TPHCM city
};
```

- **Rand_student.h:**

+ First, define vector of string class to call in functions and declare function to store first name, middle name, and last name string:

```
typedef vector<string> info_arr;
typedef vector<Student> student_arr;

static map<string, map<string, vector<string>>> addr;

//function to store first name string
static info_arr first_name()
{
    return info_arr({ "Nguyen","Ngo","Tran","Huynh","Pham","Ho","Do","Luong","Le","Phan","Vu","Vo","Bui","Duong","Ly" });
}

//function to store middle name string
static info_arr middle_name()
{
    return info_arr({ "Quang","Van","Thi","Cam","Huu","Dieu","Thoi","" });
}

//function to store last name string
static info_arr last_name()
{
    return info_arr({ "Minh","Linh","Chanh","Truong","Sang","Tai","Thinh","Phuoc" });
}
```

+ Declare setter, getter, constructor, and destructor functions in the public part of the class:

```
public: //constructor, destructor
    Rand_Student();
    Rand_Student(const student_arr&);
    Rand_Student(const Rand_Student&);
    ~Rand_Student() {}
public: //setter / getter
    Rand_Student& operator=(const Rand_Student&); //operator = function
    Rand_Student& operator+=(const Rand_Student&);//operator += function
    student_arr getArr() const;
    void setArr(const student_arr&);
```

+ Declare random functions:

```
public: //random of segments in a student's infor
    int rand_num(int, int); //random a number from min to max
    void rand_student_info(); //random student infor
    void rand_id_phone(size_t, Student&); //based on number inputed, it would  be
                                          //either random an id or random a phone series
    void rand_name_set_email(Student&); //random student's name, then convert these
                                        //characters into email format
    void rand_gpa(Student&); //random a student's GPA which in range [0.0,10.0]
    void rand_dob(Student&); //random birthdate
    void rand_address(Student&); //random address
```

+ Beside that, declare 3 funtion bunus: calculate the average GPA of all students, print out students having greater GPA than average one and sort students on their own GPA DESCENDING -> a GPA ranking list of students:

```
public:
    static double avgGpa(const student_arr&); //calculate the average GPA of all students
    void avgPrint(double); //then, print out students having greater GPA than average one
    void sortStudent(); //sort students on their own GPA DESCENDING -> a GPA ranking list of students
```

+ Declare a vector of students called _s in the private part of the class:

```
private:
    student_arr _s; //a vector of students called _s
```

- **File.h:**

+ Write function to left trim a string if left characters of string contains those chars like "\t\n\v\f\r":

```
static string& ltrim(string& str, const string& chars = "\t\n\v\f\r ")
{
    str.erase(0, str.find_first_not_of(chars));
    return str;
}
```

+ Declare function read file's data of students, write to a file with a list of students and write to a file ranking with a list of students:

```
public:
    static tuple<bool, string, student_arr> readFile(const string&);
    static tuple<bool, string> writeFile(const string&, const Rand_Student&);
    static void wRakingFile(const string&, const Rand_Student&);
```

b/ In Source file:

- **Student.cpp, File.cpp and Rand_Student.cpp:**

Write functional functions for the execution of the program as declared in the class (see the code in the sln file or cpp file)

- **MockStudentDataGenerator.cpp:**

+ Seed time and read all students saved from"student.txt" file:

```
srand((unsigned)time(0));

auto rFile = File::readFile("students.txt");
```

+ In case that file could either not be read or contain error:

-1: the file has error on being read or format definition

0: the file could not be written

1: No error, build & run successfully

```
if (!get<0>(rFile))
{
    cout << get<1>(rFile);
    return -1;
}
```

+ If be possible to read, then put these students' infor back into a vector of Student:

```
auto Students = Rand_Student(get<2>(rFile));
```

+ Create a new Student Vector in order to store randomly-generated Students' infor:

```
Rand_Student Student_Vector(student_arr(0));
```

+ Generate randomly a number of n in range of [5,10] and then generate randomly n Students:

```
int _num = Student_Vector.rand_num(5, 10);
cout << "1. Generate randomly a number n in the range of [5, 10]: " <<  _num;
```

```
cout << "\n2. Generate randomly n Students and add to the previous vector. ";
Student_Vector = student_arr(_num);
Student_Vector.rand_student_info();
```

+Add the randomly vector to the previous vector which stores a vector of Student, read from text file by operator +=

```
Students += Student_Vector;
```

+ Overwite all student saved in current vector back to "student.txt" file

```
auto wFile = File::writeFile("students.txt",Students);
```

+ As a result, output all data stored in the current Student vector:

```
cout << "\n3. Overwrite and save and the students in the current vector back to the file \"students.txt\" ";
cout << endl << endl << Students;
```

+ Print out the average GPA of all students:

```
double avgPoint = Rand_Student::avgGpa(Students.getArr());
cout << "\n4. The average GPA of all students: " << avgPoint;
```

+ Print out all the students having a GPA greater than the average GPA:

```
cout << "\n5. Print out all the students that have a GPA greater than the average GPA " << endl;
Students.avgPrint(avgPoint);
```

+ Sort Student vector DESC then print out a TOP ranking student list which is based on GPA of every each student

```
cout << "\nBONUS: \r\n      A T.O.P GPA ranking list of Sudents:    " << endl;
Students.sortStudent();
cout << Students;
File::wRakingFile("rankings.csv", Students);
```

5. Difficulty doing projects:

- The student's address part includes 3 components street, ward, district, so for random it to be objective, it needs to have a real data set.

- But each district has many wards, each ward has many streets, so the .txt file cannot be read normally, must reorganize the file.

THE END