資料結構與程式設計

(Data Structure and Programming)

106 學年上學期複選必修課程 901 31900

Homework #1 (Due: 9:00pm, Monday, Sep 18, 2017)

[Note]

(i) You need to submit this homework in order to get the 授權碼 to register the course.

(ii) Although the deadline is on Monday Sep. 18, which is the second week of the semester, you are advised to submit it at your earliest convenience in order to get the 授權碼 earlier.

(iii) TA or instructor will quickly go over your submission of homework and grade it ASAP. Plagiarism will be checked. You must pass the plagiarism check and have a "reasonable" submission in order to qualify for the 授權碼. (Note) "Reasonable" is a qualitative judgment, not a quantitative measurement, meaning that you should not turn in blank homework or something that is not abide by our requirements. Homework score is not a final decision factor for it.

(iv) Please do pay attention to the homework submission rules at the end of this document. Failure to follow the rules will result in disqualification of getting 授權碼.

(v) It is very likely you are NOT familiar with some advanced C++ features required in this homework. For example, dynamic array, operator overloading, or even header file / multiple files, etc. Nevertheless, "self-learning" by researching on the internet and then figuring out the solution is one of the key spirit in this course. You need to get used to it in order to enjoy (a.k.a. survive in) the class.

(vi) If you find it very difficult in finishing this homework, you may consider NOT to take this course as the followed homework assignments will become much tougher.

(vii)

---

1. In this course, we require that all the homework be done in a Linux compatible programming environment (e.g. Ubuntu, Mac OS X, etc). Therefore, we would like to make sure you have it before the class starts.

   There are several ways to obtain a Linux compatible environment, for example: (i) Have a computer that runs Linux OS, (ii) Have a dual-boot computer that you can choose to boot Linux when powering it on, (iii) Install a virtual machine (e.g. VirtualBox) on a non-Linux environment (e.g. Windows) and then install the Linux OS on the virtual machine, (iv) Remotely login (i.e. telnet or ssh) to a Linux workstation (e.g. some server in your lab), or (v) Use "terminal" on Mac, etc.

   In this problem, we will check if you have "g++" installed and if it supports C++11. please follow the steps below, screenshot your "terminal", rename and save the file under the p1 directory.

   (i) In terminal, execute "cd p1" // change directory to "p1"

   (ii) Execute "make" // it will evoke the "Makefile"

   If it is executed correctly, you should see:

   ```
   Compile success!!
   program tested successfully !!
   compiler check pass !!
   ```

   If you see any problem on C++11 compatibility or g++ compiler, fix it before you move on.

   (iii) Screenshot your terminal, and save the screenshot as "p1.xxx", where "xxx" is the file extension of the image (e.g. p1.jpg).

   (iv) Make sure "p1.xxx" is stored properly under the p1 directory.

   Feel free to mosaic parts of the screenshot if you have any security concern.

   Execute the above steps in the same terminal so that the screenshot can be fitted into a single image file.

2. In this problem, you are asked to read in a ".csv" file and perform some operations and/or statistics on its data.

   A .csv file is commonly used to represent a table of data where cells are separated by comma ',' and rows are separated by a carriage-return symbol (i.e. ctrl-m, `^M`, '\r', ASCII code = 13) and/or a line-feed/new-line symbol (i.e. ctrl-j, `^J`, '\n'). Please note that all the .cvs files provided and tested in this homework are generated by the Microsoft Excel Program. However, the occurrence of `^M` and `^J` may vary depending on the platform/tool the .csv file is generated.

   For example, for the following table ---

1	2
3	
5	6
	8

   Saving it as a .csv file, and you may find its content as:

   ```
   1,2^M^J3,^M^J5,6^M^J,8
   ```

No extra character, except probably another `^M and/or ^J`, is presented after the last data. Therefore, you should be able to calculate the number of rows by the apparence of `^M and/or ^J` and the number of columns by the occurrences of ‘,’. (That is, the numbers of rows and columns are NOT given)

Note that: (1) All the cells, if not empty, in the table contain **integral** data with the range from `INT_MIN` to `INT_MAX` – 1 (as defined by the type “int” in C++). You don’t need to handle other data type. For empty cell, you can store it as `INT_MAX`. However, when it is printed, you should print it as ‘.’, not `INT_MAX` nor its corresponding integral value (see `PRINT` command). (2) The number of columns won’t be changed after the table is read in. However, user may add new rows to the table and thus the number of rows may increase. (3) Data won’t be removed once it is read into the table. (4) You can also assume that the tested .cvs files and commands are all in correct format. That is, you don’t need to handle syntax errors in reading .csv file (e.g. non-integral symbol) or parsing commands. (5) You also don’t need to handle any computing errors (e.g. integer overfloat, divided by zero, etc) for the operations of the commands.

Implement the following sub-problems using C++ in the subdirectory “p2” and write your codes in the files as specified in each sub-problem. To compile the codes, type “make” under “p2” directory to evoke “Makefile” to generate the executable “p2Run”. In principle, you don’t need to create new “.h” or “.cpp” files. However, if you intend to do so, please make sure you understand how to incorporate the new files into the Makefile and the executable can be successfully generated by “make”.

Under the success of compilation, type “./p2Run” to run the program.

(a) Define the *main()* function in “p2Main.cpp” and declare a table “`Table table`” in it.

In the beginning, your program should prompt the message to ask for the input of the .csv file.

```
Please enter the file name: test1.csv
```

Once the .csv file is properly read in, print the message (change the filename accordingly):

```
File “test1.csv” was read in successfully.
```

and leave the cursor on the screen waiting for the user to enter any of the commands as defined in sub-problems (d) – (g). Since we assume that there is no syntax error in the .csv file, there is no need to handle the parsing error. Therefore, if your program does not halt and print out the above message, it is definitely something wrong in your code.

Please note that the numbers of columns and rows are not specified, you should figure them out on your own when parsing the .csv file.

(Hint) Suggested readings: iostream, fstream, string, getline(istream&, string&)

(b) Define `class Row` and `class Table` in the file "p2Table.h" to represent the data of a row and the table, respectively. Since the number of elements in a row (i.e. the number of columns in the table) is not known in the beginning and is fixed after the .csv file is read in, please use dynamic array "`int *`" to allocate the memory dynamically. That is,

```
class Row {
public:
   // TODO: define constructor and member functions on your
own

   // Basic access functions
   const int operator[] (size_t i) const { return _data[i]; }
   int& operator[] (size_t i) { return _data[i]; }

private:
   int  *_data; // DO NOT change this definition. Use it to store data.
};
```

In other words, once the number of columns (say, '`s`') is calculated from the .csv file, you can construct this data member as a dynamic array like "`_data = new int[s]`".

Note that the overloaded operator [] is to access the i^{th} element in "`_data`". There is a non-const version (i.e. return by reference) so that users can write data into the object using the [] operator (e.g. row[3] = 5). Please feel free to define more member functions in order to operate on the data. DO NOT change "`_data`" to other data type (other than int *) and to *public*, and DO NOT use "`friend`" to bypass the data encapsulation.

(Hint) Suggested readings: class / constructor, dynamic array, operator overloading, return by reference, const method/variable

(c) The `class Table` is to represent the data table. Since the number of rows can be increased on demand, please use STL dynamic array "`vector<Row> _table`" as its data member. That is,

```
class Table {
public:
   // TODO: define constructor and member functions on your
own
   bool read(const string&); // called in main()

   // Basic access functions
   size_t nCols() const { return _nCols; }
   const Row& operator[] (size_t i) const { return _rows[i]; }
   Row& operator[] (size_t i) { return _rows[i]; }
private:
   size_t      _nCols; // You should record the number of
columns.
   vector<Row>  _rows; // DO NOT change this definition.
                       // Use it to store rows.
};
```

With such design, to access the i^{th} row in table, you can use "`table[i]`", and to obtain the value of the j^{th} cell of the i^{th} row, you can use "`table[i][j]`".

(Hint) Suggested readings: STL vector

For the commands in sub-problems (d) – (g) below, implement them as member functions of `class Table` and name them as "`print()`", "`add()`"… etc. accordingly. Evoke them in "p2Main.cpp" such as `table.print()`. Note that the commands are case sensitive (all uppercase letters) and you don't need to handle errors when parsing the commands/options.

(d) The command "`PRINT`" prints the entire table content in a tabular format. For example, for the example in the previous page, you should see:

```
1   2
3   .
5   6
.   8
```

Note that empty cells are printed as '.'. Please use "`setw(4)`" and "`right`" for "`cout`" to control the output format.

(Hint) Suggested readings: #include <iomanip>

(e) The commands "SUM", "AVE", "MAX", "MIN" and "DIST" compute the summation, average, maximum, minimum, and distinct count (i.e. the different values) of data in the specified column. Clearly, there will be only one argument (i.e. the column index) for these commands. Null cells are ignored (not treated as '0') here. Note that column index starts from '0' (the first column).

For example, for the following table (i.e. "test2.csv")---

7	2	5	5	8	
3		6	7		10
5	6		4	9	0
	8	4		7	
7		5	9		1
8		3	0	4	1
	4	4	4		4

```
SUM   3   // compute the summation of data in column #3.
MAX   0   // print the maximum data in column #0.
MIN   4   // print the minimum data in column #4.
DIST  2   // count the number of distinct data in column #2.
```

The output of these commands should be as the following ---

```
The summation of data in column #3 is 29.
The maximum of data in column #0 is 8.
The minimum of data in column #4 is 4.
The distinct count of data in column #2 is 4.
```

For the `AVE` command, compute and round the result up to the first decimal digit (Use `fixed` and `setprecision(1)` to format the printing). For example, entering "`AVE 3`" and "`AVE 0`", you will get:

```
The average of data in column #3 is 4.8.
The average of data in column #0 is 6.0.
```

If there is an empty column (i.e. all the cells in this column are empty), for the above commands, you should report:

```
Error: This is a NULL column!!
```

(f) The command "ADD" adds a new row to the end of the table. The added data are provided as arguments to this command and are separated by space. For the column with intended null data, put a '.' symbol to represent it. Therefore, there must be exactly *#columns* arguments for this command.

For example ---

```
ADD 9 10   // add a new row with data  9, 10.
ADD . 12   // add a new row with data   , 12.
ADD -3 .   // add a new row with data -3,   .
```

Nothing will be printed for this command. Use "PRINT" command to check the result.

(g) The command "EXIT" exits the program.

Notes: Please pay attention to the homework rules below and announcements on the website/FB page. Failure to abide by the rules may result in deduction of your homework score.

[Homework Rules]

1. Name your files as specified in each problem. Put all your files and in a directory named "yourID_hw1", where "yourID" is your school ID in lower case and numbers (e.g. b04901888_hw1). Compress the directory by the command (on Linux or Mac Terminal):

   ```
   tar zcvf yourID_hw1.tgz yourID_hw1
   ```

   You should remove the executable and object files (if any) before submission to reduce the file size (e.g. type "make clean" under p2).

2. Submit your homework through ftp. The instruction of ftp will be sent out separately in a few days.

3. Any update, problem fix, and announcement will be posted on FaceBook group NTU_DSnP: https://www.facebook.com/groups/NTUDSnP/. Please follow it closely. If you are a registered student and would like to join the group, please go there and click "Join".