CS251 - Project 1: C++/Java Programming Example

Out: Jan/14/2019, 8am
Due: Jan/21/2019, 8am

1. Overview

Write a simple program that reads in an example file and outputs a file according to the given requirements. The project also serves to familiarize you with the programming environment you will use during the semester.

2. Detailed Description

You are going to write a program that reads a text file of non-negative integers and outputs a text file containing numbers arranged in a triangular format.

The input file has the following format:

- It has two lines.

- All integers are delimited by a single white space and there is no white space at the end of a line.

- Line 1 has two integers: N and X.
  N: specifies the maximum number of digits that each number in the line 2 can have. For example, if N = 3, then only numbers from 0-999 will be included in the input file.
  X: is a value representing the increment in numbers per line in the output file. The first line of numbers in the output file always has only a single number. If X=3, then the second line of numbers should have 1+3=4 numbers, the third line will have 4+3=7 numbers, and so forth. If X=2, then the first line of numbers should have only a single number, second line of numbers have 1+2=3 numbers, third line of numbers should have 3+2=5 numbers, and so forth.

- Line 2 has a sequence of numbers that should be printed in the same order when read from top-bottom and left-right in the output file but as per the format described above.

- We guarantee the input file to be of a *correct format*.

The output should have the following format:

- All numbers should be placed between pipe symbols "|".

- If the maximum number of digits is N but a number you read from the second line of the input file has digits less than N (e.g. 1, 2, 5, 10, 13 when N=3), you should print one **extra** white space in the output file for each missing digit.

- The number between the pipe symbols should also have one white space before and after (in addition to any white space added to ensure each number has N digits).
- The number between the pipe symbols should be left-justified – this implies that all numbers have one white space between a pipe symbol and the first digit of the number and one or more white spaces between the last digit of the number and a pipe symbol.
- There should be no white spaces before the first pipe symbol on a line nor after the last pipe symbol on a line.
- Sample input files and output files are provided to you.

For example:

Input-1:
```
2 1
1 32 13 13 4 54
```

Output-1 should be:
```
| 1  |
| 32 | 13 |
| 13 | 4  | 54 |
```

Input-2:
```
1 2
1 2 3 4 5
```

Output-2:
```
| 1 |
| 2 | 3 | 4 |
| 5 |
```

Input-3:
```
1 0
1 2 3 4 5
```

Output-3:
```
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
```

3. Grading and Programming Environment
Assignments will be tested in a Linux environment. You will be able to work on the assignments using the Linux workstations in HAAS and LAWSON (with your CS username and password). You are provided with two sample input files; however, in addition to these sample files your program will be tested on some withheld test files.

3A. Java

- Compiler we use: javac (v10.0.2)
- File to submit: TriangularNums.java **only.**

Your project must compile using the javac compiler (v10.0.2) on data.cs.purdue.edu. You **only** need to submit **TriangularNums.java** which is located under src/. You need to implement the **constructor** of TriangularNums and the function **writeToFile**(String) in TriangularNums.java. Main.java is provided to you so that you can generate an output file on your own to see if your code works correctly; you may change it as your wish. We will run JUnit tests which look like test/TestSolution.java to grade your work. However, not all grading test cases are provided in test/TestSolution.java. You may write your own JUnit test cases in test/TestSolution.java.

Compiling process:

- To compile Main:
    1. Go to the project root.
    2. Type command in console: javac src/*
- To run Main:
    1. Go to the project root.
    2. Type command in console: java -cp src/ Main InputFilePath OutputFilePath
       Note: You should replace InputFilePath OutputFilePath with actual file path in the above command.

- To compile JUnit Tests:
    1. Go to the project root.
    2. Type command in console: javac -cp src/:lib/* test/TestSolution.java
- To run JUnit Tests:
    1. Go to the project root.
    2. Type command in console:
       java -cp src/:lib/*:test/ org.junit.runner.JUnitCore TestSolution

Grading process:

1. Compiling your program with JUnit test files using javac.
2. Running your program on several JUnit test cases. These test cases basically create TriangularNums instances by using different pre-made input files, then generate output files through calling writeToFile() function. Finally it checks if the output file generated by your code matches the expected output. Any forms of

mismatch (including extra white spaces, line separators) can cause point deduction.

3B. C++
Your project must compile using the standard g++ compiler (v7.3.0) on data.cs.purdue.edu. For convenience, you are provided with a template Makefile and C++ source file. You must submit all the source code as well as the Makefile that compiles your provided source code into an executable named “project1-sol”. (Note that the file name is in lowercase and does not have a file extension).

The grading process consists of:

1. Compiling and building your program using your makefile.
2. Running your program automatically with several test input files we have pre-made according to the strict input file format of the project and verifying correct output files. Thus follow the above instructions for generating the output precisely **– do not “embellish” the output with additional characters or formatting** – if your program produces different output such as an extra prompt, characters, and/or white space, points may be deducted. (It is fine however if you print your extra output to stderr if it helps in debugging.)

Your program will be run with two command line arguments. The first argument would be the name of input file and second argument would be the name of the output file:

project1-sol input-test1.txt output-test1.txt

1. *project1-sol* is name of the executable program.
2. The program should read input from a file named *input-test1.txt*
3. The program should create a file name *output-test1.txt,* and write the correctly formatted output to it.

Important:
1. If your program does not compile, you get an automatic 0.
2. Plagiarism and any other form of academic dishonesty will be graded with 0 points as definitive score for the project and will be reported to the corresponding office.

Submission Instructions:
The homework must be turned in by the due date and time using the turnin command. Follow the next steps:

1. Login to data.cs.purdue.edu (you can use the labs or a ssh remote connection).

2. Make a directory named with your username (in lowercase) and copy in the files you want to submit.

3. Go to the upper level directory and execute the following command:
   turnin -c cs251 -p project1 your_username
   (**Important:** previous submissions are overwritten with the new ones. Your last submission will be the official one and therefore graded).

4. Verify what you have turned in by typing **turnin -v -c cs251 -p project1** (**Important:** Do not forget the **-v** flag, otherwise your submission would be replaced with an empty one).

** Please PLAN AHEAD. Attend your PSO and come to TA office hours if you have problems! **