The University of Melbourne
Department of Computing and Information Systems
COMP20005 Engineering Computation
Semester 2, 2016
Assignment 1
Due: 4:00pm Wednesday 21st September 2016

1 Learning Outcomes

In this assignment you will demonstrate your understanding of loops and `if` statements by writing a program that sequentially processes a file of text data. You are also expected to make use of functions; and in Stage 3 of the project, you will need to use strings.

2 The Story...

Theatrical box office earnings are a relatively objective metric for evaluating how successful a movie is. Statistics on the theatrical box office earnings help the movie industry understand the different factors contributing to the success of a movie. For example, from the list the high-grossing films[1], it is observed that war films (e.g., *Top Gun*), musicals (e.g., *The Lion King*) and historical dramas (e.g., *Titanic*) have been popular in the last century. In the 21^{st} century, franchise movies (e.g., the *Harry Potter* series) have dominated the list.

Below is a list of high-grossing movies, where the five columns represent the unique ID, year of release, gross earnings (in millions), budget (in millions), and title of a movie, respectively. Among the ten movies listed, *Avatar* has the highest gross earnings (over 2700 millions). An observation is that, a high budget does not always guarantee a high gross-budget ratio. The movie *Pirates of the Caribbean: At World's End* is an example. It has the highest budget among the ten movies, but its gross-budget ratio is the lowest ($963/300 = 3.21$).

```
1  2007 963  300 Pirates_of_the_Caribbean_At_Worlds_End
2  2008 1004 185 The_Dark_Knight
3  2009 2787 237 Avatar
4  2010 1000 50  A_Special_Example_Movie
5  2011 1341 250 Harry_Potter_and_the_Deathly_Hallows_Part_2
6  2012 1519 220 The_Avengers
7  2013 1287 150 Frozen
8  2014 1104 210 Transformers_Age_of_Extinction
9  2015 2068 200 Star_Wars_Force_Awakens
10 2016 1152 250 Captain_America_Civil_War
```

A less explicit factor to consider is inflation. In the past ten years, an average of 1.81% inflation has been recorded in the US[2]. Taking it into consideration, gross earnings of 963 million in 2007 projects to 1131 million in 2016 ($1131 = 963 \times (1 + 0.0181)^{2016-2007}$).

[1] `https://en.wikipedia.org/wiki/List_of_highest-grossing_films#High-grossing_films_by_year`
[2] `http://www.usinflationcalculator.com/inflation/consumer-price-index-and-annual-percent-changes-from-1913-to-2008/`

3 Your Task

In this assignment, you will be given a list of movie gross earnings data like the one shown above. Particularly, each line of the list contains the information of one movie separated by "tabs", including the unique ID (an integer in the range of $[0, 99]$), the year of release (an integer in the range of [1900, 2016]), the gross earnings (an integer in the range of [1, 3000]), the budget (an integer in the range of [1, 1000]), and the title (a string formed by up to 100 characters, which may be letters, digits, or underscores '_'). *You may assume that the list will always be correctly formatted and contain at least 1 and at most 99 movies.*

Your task is to write a program that reads the list and output statistics calculated on it. The assignment consists of the following four stages (Stage 4 is for a challenge and is optional).

3.1 Stage 1 - Processing the First Movie (Marks up to 5/10)

You can complete this stage without using strings. However, you may also use strings if you are confident that you will be proceeding to Stage 3.

Write a program that reads the first line of the input data, and prints out for the first movie: the ID (2-digit integer), year of release (4-digit integer), gross earnings (4-digit integer), budget (4-digit integer), gross-budget ratio (gross/budget, 2 digits both before and after the decimal point), and the projected gross earnings (to the year 2016, 4-digit integer) calculated based on the following equation:

$$projected\_gross = \lfloor gross\_earnings \times (1 + avg\_inflation\_ratio)^{2016 - year\_of\_release} \rfloor.$$

Here, "$\lfloor\rfloor$" means to "take the integer part of" (*no rounding*). For simplicity, we assume that $avg\_inflation\_ratio$ has a constant value of 0.0181.

Hint: You may use the `pow()` function provided by the `math.h` library for the power calculation.

The output of this stage given the above sample input should be (where "`mac:`" is the command prompt):

```
mac: ./assmt1 < test0.txt
Stage 1
==========
Movie 01 (2007) grossed in 0963m with a budget of 0300m
Gross-budget ratio: 03.21
Projected gross: 1131m
```

As this example illustrates, the best way to get data into your program is to edit it in a text file (with a "`.txt`" extension, `jEdit` can do this), and then execute your program from the command line, feeding the data in via input redirection (using `<`). In your program, you will still use the standard input functions such as `scanf()` to read the data fed in from the text file. Our auto-testing system will feed input data into your submissions in this way as well. To simplify the assessment, your program should **not** print anything except for the data requested to be output (as shown in the output example).

3.2 Stage 2 - Processing the Rest of the Movies (Marks up to 7/10)

You can complete this stage without using strings. However, you may also use strings if you are confident that you will be proceeding to Stage 3.

Now modify your program so that the gross-budget ratio values of all of the input movie data are computed and visualised. You may assume that the gross-budget ratio is within the range of $(0, 100)$. On the same sample input data the additional output for this stage should be:

```
Stage 2
==========
Movie 01, gross-budget ratio: 03.21 |----
Movie 02, gross-budget ratio: 05.43 |------
Movie 03, gross-budget ratio: 11.76 |---------+--
Movie 04, gross-budget ratio: 20.00 |---------+---------+
Movie 05, gross-budget ratio: 05.36 |------
```

```
Movie 06, gross-budget ratio: 06.90 |-------
Movie 07, gross-budget ratio: 08.58 |---------
Movie 08, gross-budget ratio: 05.26 |------
Movie 09, gross-budget ratio: 10.34 |---------+-
Movie 10, gross-budget ratio: 04.61 |-----
```

You need to work out how the visualisation works based on this example. Note that you should *write functions* to process this stage where appropriate.

3.3 Stage 3 - Overall Reporting (Marks up to 10/10)

To complete this stage you will need to use strings to store the movie titles. If you have completed Stages 1 and 2 without strings, you should save a copy of that version of your program into a separate file, as an insurance policy that can be submitted again later if you are unable to get your Stage 3 solution operational.

The additional output from this stage is the total number of movies, the title of the movie with the highest projected gross earnings, and this highest projected gross earnings. In the case of ties of projected gross earnings, the movie with the smallest ID should be output.

```
Stage 3
==========
Total: 10 movies
Highest projected gross movie: Avatar
Highest projected gross: 3159m
```

Hint: You do not need to use an array of strings. But you will need to use one or more strings to correctly compute the required output. Wherever appropriate, code should be shared between the stages through the use of functions. In particular, there should *not* be long stretches of repeated code appearing in different places in your program. Further examples showing the full output that is required are provided on the LMS.

3.4 Stage 4 - For a Challenge (and Not for Submission)

For a challenge, try printing out the movies sorted in the descending order of their gross-budget ratios. *But please don't submit these programs.*

4 Submission and Assessment

This assignment is worth 10% of the final mark. A detailed marking scheme will be provided on the LMS.

You need to submit all your code in one file named `assmt1.c` for assessment; detailed instructions on how to submit will be posted on the LMS once submissions are opened. Submission will NOT be done via the LMS; instead you will need to log in to a Unix server and submit your files to a software system known as `submit`. You can (and should) use submit both **early and often** - to get used to the way it works, and also to check that your program compiles correctly on our test system, which has some different characteristics to the lab machines. Only the last submission made before the deadline will be marked.

You will be given a sample test file `test0.txt` and the sample output `test0-output.txt`. You can test your code on your own machine with the following command and compare the output with `test0-output.txt`:

```
mac: ./assmt1 < test0.txt    /* Here '<' feeds the data from test0.txt into assmt1 */
```

You may discuss your work with others, but what gets typed into your program must be individual work, **not** copied from anyone else. Do **not** give hard copy or soft copy of your work to anyone else; do **not** "lend" your memory stick to others; and do **not** ask others to give you their programs "just so that I can take a look and get some ideas, I won't copy, honest". The best way to help your friends in this regard is to say a very firm "no" when they ask for a copy of, or to see, your program, pointing out that your "no", and their acceptance of that decision, is the only thing that will preserve your friendship. *A sophisticated program that undertakes deep structural analysis of C code identifying regions of similarity will be run over all submissions in "compare every pair" mode.* See https://academichonesty.unimelb.edu.au for more information.

Deadline: Programs not submitted by **4:00pm Wednesday 21st September 2016** will lose penalty marks at the rate of two marks per day or part day late. Late submissions after 4:00pm Friday 23rd September 2016 will **not** be accepted. Students seeking extensions for medical or other "outside my control" reasons should email the lecturer at jianzhong.qi@unimelb.edu.au. If you attend a GP or other health care professional as a result of illness, be sure to take a Health Professional Report form with you (get it from the Special Consideration section of the Student Portal), you will need this form to be filled out if your illness develops into something that later requires a Special Consideration application to be lodged. You should scan the HPR form and send it in connection with any non-Special Consideration assignment extension requests.

And remember, *C programming is fun!*