Project Descriptions:

In this project, we are going to develop three programs as specified below using PC-SPIM (MIPS assembly instructions):

(1) IF ~ THEN ~ ELSE structure
(2) WHILE structure
(3) SWITCH~CASE structure

The descriptions and requirements for each control structure are as follows:

(1) IF ~ THEN structure

When your program starts, your program prompts a user to enter two numbers. The first number is for your course score (in the 100 point scale), which should take an integer in the range of 0 through 100 (0 and 100 are included). The second number is the “curve value”, which should be in the range of 2 through 8 (2 and 8 are included). The two numbers should be prompted by your program, one at a time (the course score, followed by the curve value). When a user enters an invalid number, your program should detect it and repeat the prompt to a user. When a user made an invalid input to the curve value, your program should prompt only for the curve value, but not for the course score. After the two numbers are successfully entered, your program should determine a course letter grade for the user, based on the following grade criteria:

<table>
<thead>
<tr>
<th>Score Range</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-92:</td>
<td>A</td>
</tr>
<tr>
<td>91-82:</td>
<td>B</td>
</tr>
<tr>
<td>81-72:</td>
<td>C</td>
</tr>
<tr>
<td>71-62:</td>
<td>D</td>
</tr>
<tr>
<td>Below 62:</td>
<td>F</td>
</tr>
</tbody>
</table>

For example, if a user earned 90 for his (her) course score, and the curve value is 2, the user should get ‘A’ for the course (and the same logic is applied to the four other letter grade categories).

Requirements:

a. The above procedure must be implemented in MIPS assembly instructions.
b. Each input must be tested to confirm its validity
c. If the first input (a course score) is correctly made, the first input should never be repeated when a user makes an invalid input for the second input (i.e., “the curve value”).
d. Your program should output a correct letter grade (no garbage output, and etc.). See the following screen snapshot of the program.
e. Your program should terminate without any error or a warning message.
(2) WHILE structure

This program displays a certain number (3 to 30) of Fibonacci numbers\(^1\). When your program starts, your program prompts a user to enter how many Fibonacci numbers the user would like to calculate. If a user makes an invalid input (i.e., less than 3 or larger than 30), then the program should prompt the user to make a valid input after it displays an error message (as shown below). This process should be repeated until the user makes a valid input.

When a user makes a valid input, the program should calculate and display the Fibonacci numbers using a loop structure implemented in MIPS assembly instructions.

\[^1\] Fibonacci numbers are the sequence of the numbers that are generated as \(F_{i+2} = F_{i+1} + F_i\), where \(i = 0, 1, 2, \ldots \infty \) and \(F(0) = F(1) = 1\).

Requirements:

a. The above program should be implemented using a loop structure implemented in MIPS assembly instructions.
b. Each input must be tested to confirm it is a valid input.
c. If an input number is less than 3, display an error message, “the entered number is less than 3”. Then, prompt a user to make another input.
d. If an input number is larger than 30, display an error message, “the entered number is larger than 30”. Then, prompt a user to make another input.
e. Your program should correctly calculate the Fibonacci numbers.
f. Each Fibonacci number is displayed after a number count (see the figure above).
g. Your program should display the Fibonacci numbers, exactly as many as specified by a user.
h. Your program should terminate without any error or a warning message.

(3) SWICTH ~ CASE ~ structure

This program first prompts a user to enter two numbers: this year (2020 - 2120) and your age (0 – 110). The two numbers should be prompted by your program, one at a time (this year and your age, in that order) after the program starts. When a user enters an invalid number, your program should detect it and repeat the prompt to a user. When a user made an invalid input for his/her age (after the user correctly enters this year), your program should prompt only for the user’s age, but not for this year. After the two numbers are successfully entered, your program identifies the animal of the year in Chinese Zodiac*2 and display the comments for the animal. The comments to be displayed are posted to CS286 course home and you can use the messages by “copy and paste” to your program source code file.


The suggested program logic for Program #3:  

Step 1: Calculate the year you were born by:
The year you were born = (this year) – (your age)

Step 2: Subtract 1900 from the year you were born. Call the result ‘X’.
X = (the year you were born) - 1900

Step 3: Repeatedly subtract 12 from X until we can no longer subtract 12 and call the remainder ‘Y’.

Step 4: Identify the animal of yours as:
If ‘Y’ = 0 → you are rat.
If ‘Y’ = 1 → you are ox.

Step 5: Display a message for the animal you belong to.

Requirements:

a. The above program should be implemented using MIPS assembly instructions.
b. If an invalid input is made, an error message should be displayed and your program should prompt a valid input.
c. After the first input (this year) is successfully made, the first input should never be prompted, especially when an invalid input is made to the second input (your age).
d. The response for each input should be correctly displayed.
e. Your program should terminate normally (should not crash the system when your program leaves the system or should not produce any garbage outputs).

Submissions:

Your source code files (one *.asm source code file for each of the three programs) should be e-mailed to the T.A. by 9:30 A.M. on February 11th.

- Extra-Credit (+3 points to your midterm exam score) Early-Submission* Deadline: 11:59:59 P.M., February 3rd.

Note 1: Each program flow-control should be implemented as an independent program (each of you is expected to submit three *.asm source code files).

Note 2: In your program source code, include your name and student ID in your program header (at the beginning of your *.asm source code file).

Note 3: It is required to use PC-SPIM posted to the CS286-001 course homes (programs developed for different versions of PC-SPIM will NOT be accepted for grading).

Note 4: This programming project is an individual programming project (no collaboration with others to complete the programming project is allowed). Violation of this rule is considered academic dishonesty.

Note 5: Academic dishonesty in Project #1 will at least result in 0 point for the entire programming project credit for this course (not only for Project #1). Serious academic dishonesty will result in a failing grade (‘F’ grade) for repeated violations.
in this course and the names of people will be reported to the CS department and the School of Engineering Dean’s Office.

Late Submission:

- Penalty of -10% will be given for every 24 hours after the due (i.e., -10% for a submission within the first 24 hours after the due).
- Submission more than 48 hours after the due will not be accepted.

*3: No “late submission” for early deadlines. The extra credit will be given only if all the three programs are completed by the extra-credit early submission deadline. No “partial extra credit” will be given for this extra credit opportunity.

Grading Criteria:

- **Assembler error:** If Dr. Fujinoki can not run your assembly source code file due to any assembler error: 0 ~ 20% of the credit for this assignment phase will be given.

- **Run-time error:** After your assembly source code file is successfully assembled to machine codes, if your program does not satisfy any requirement: 5% penalty for each minor problem. For any major failure, penalty depends on each such major problem.

- **The weight** of the three programs is: 30%, 35%, and 35% for program 1 (if-then), program 2 (while), and program 3 (switch-case) respectively.

Expectations when a question is asked:

1. Identify where (in your source code) the problem exists
2. Describe the symptom(s) of the problem
3. Describe how the problem happens (always happen, sometime happen, the condition(s) for the problem to happen, etc.)
4. Describe what you tried (to understand and/or solve the problem)
5. Stop by Dr. Fujinoki’s office (no question through e-mail).
6. Bring your source code (either hard copy or soft copy)