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

Illinois’s basic speeding law prohibits driving at a speed that is “greater than is reasonable and proper with regard to traffic conditions and the use of the highway, or endangers the safety of any person or property.” In other words, motorists must always drive at a safe speed. What a safe speed is will depend on the circumstances (625 Ill. Comp. Stat. Ann. § 5/11-601 (2017)).

When your program starts, your program prompts a user to enter two numbers. The first number is your current driving speed in MPH (1 to 200 MPH, where ‘1’ and ‘200’ are included). The second number is the absolute speed limit specified for the road you are currently running on (15 to 70 in MPH, where ‘15’ and ‘70’ are included). The two numbers should be prompted by your program, one at a time (your current driving speed, followed by the speed limit). When a user makes any invalid input, your program should detect it and repeat the prompt to a user. When a user made an invalid input to the speed limit (the second input), while the user had made a valid input to the first (i.e., your current driving speed), your program should prompt only for the second input, but never for the first input. After the two numbers are successfully entered, your program should determine the penalty you may receive based on the following criteria (730 Ill. Comp. Stat. §§ 5/5-4.5-55, 5/5-4.5-60 (2017)):

- Exceeding the speed limit by 1-20 MPH over the limit: “$120 fine”
- Exceeding the speed limit by 21-25 MPH over the limit: “$140 fine”
- Exceeding the speed limit by 26-34 MPH over the limit: “Class B misdemeanor and carries up to six months in jail and a maximum $1,500 in fines”.
- Exceeding the speed limit by 35 MPH or more over the limit: “Class A misdemeanor and carries up to one year in jail and a maximum $2,500 in fines”.

For example, if you are driving at 72MPH on a highway with 55MPH limit, a message: “you may receive a $120 fine” should be displayed on your local monitor. If your current speed is less than the speed limit, a message “you are a safe driver!” should be displayed. For each invalid input, “You made an invalid input for X. Enter a valid input for X.”, where X is:

For the first input: “your current driving speed”
For the second input: “the speed limit for the road”

For Program #1, the following MIPS (PC-SPIM) floating-point instructions and system calls are suggested:\*: 

- **li** – load a constant signed two’s complement integer to a register
- **la** – load a constant unsigned integer (usually memory address) to a register
- **move** – copy the content of a register to another register
- **sub** – subtract the content of a register from the content of another register and save the result to a destination register
- **j** – let the processor jump to the instruction specified by “jump destination address”
- **beq, bgt, blt**, and etc. – conditional branch instructions for controlling flow of the program
- **PC-SPIM system call #4** – output (print) a pre-declared text string to the local display
- **PC-SPIM system call #5** – input a signed (negative and positive) integer from the keyboard

\*: a reference to those instructions and system calls are available in Appendix-B in the required textbook.

Requirements:

- The above procedure must be implemented using IF-THEN structures implemented in MIPS assembly instructions.
- Each input must be tested to confirm its validity.
- After 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 speed limit of the road”).
- Your program should output a correct message (no garbage output, and etc.). See the following screen snapshot of the program.
- Your program should terminate without any error or a warning message.

```
> Console
Enter your current driving speed in MPH (1 to 200): 0
You made an invalid input for your current driving speed. Enter a valid input for your current driving speed
Enter your current driving speed in MPH (1 to 200): 24
You made an invalid input for your current driving speed. Enter a valid input for your current driving speed
Enter your current driving speed in MPH (1 to 200): 62
You made an invalid input for your current driving speed. Enter a valid input for your current driving speed
Enter your current driving speed in MPH (1 to 200): 10
You made an invalid input for the absolute speed limit. Enter a valid input for the speed limit.
Enter the absolute speed limit specified for the road you are currently running on (15 - 70): 4
You made an invalid input for the absolute speed limit. Enter a valid input for the speed limit.
Enter the absolute speed limit specified for the road you are currently running on (15 - 70): 88
You made an invalid input for the absolute speed limit. Enter a valid input for the speed limit.
Enter the absolute speed limit specified for the road you are currently running on (15 - 70): 88

You may receive a $120 fine
```
(2) WHILE structure

Develop a *.asm program (for PC-SPIM) that generates (calculates) a sequence of the numbers based on the following rules:

- Your program takes the following three parameters
  
  (a) The origin of a number sequence (as an integer) between 0 and 7 (i.e., no less than 0 and no greater than 7).

  (b) There should be two addition factors, called “Delta 1” and “Delta 2” (as an integer), each should be between 2 and 9 (i.e., no less than 2 and no greater than 9).

  (c) The total number of the numbers to be generated (as an integer): between 4 and 30 (i.e., no less than 4 and no greater than 30).

- Then, your program generates the number sequence (and display them on the local monitor) by adding “Delta 1” to “the origin number” you entered for (a) above. Then, the next number should be calculated by adding the value of “Delta 2” to the number calculated by adding “Delta 1” to “the origin number”. After that, numbers should be generated by alternatively add “Delta 1” or “Delta 2” to the previous number (as shown below).

- At the end of your program run, the check-sum (the sum of the numbers displayed by your program) should be displayed.

```
Console
Enter the origin of your number sequence (0 - 7): 6
An invalid input is detected for the origin of your number sequence. Please try it again.
Enter the origin of your number sequence (0 - 7): -3
An invalid input is detected for the origin of your number sequence. Please try it again.
Enter the origin of your number sequence (0 - 7): 4

Enter your DELTA-1 (2 - 9): 1
An invalid input is detected for your DELTA-1. Please try it again.
Enter your DELTA-1 (2 - 9): 10
An invalid input is detected for your DELTA-1. Please try it again.
Enter your DELTA-1 (2 - 9): 2

Enter your DELTA-2 (2 - 9): 13
An invalid input is detected for your DELTA-2. Please try it again.
Enter your DELTA-2 (2 - 9): -3
An invalid input is detected for your DELTA-2. Please try it again.
Enter your DELTA-2 (2 - 9): 5

Enter the total number of the numbers (4 - 30): 2
An invalid input is detected for the the total number of the number. Please try it again.
Enter the total number of the numbers (4 - 30): 33
An invalid input is detected for the the total number of the number. Please try it again.
Enter the total number of the numbers (4 - 30): 6
4, 6, 9, 11, 14, 16

Check-sum: 60
```

Requirements:

1. Any invalid input should be detected.

2. When an invalid input is detected, the input should be repeated until a valid (acceptable) input is made by a human user.
3. When an invalid input is detected for any of (a), (b), (c), and (d), the input prompt should NOT be repeated for any input that has been correctly made (e.g., when an invalid input is detected for (b) above after a valid input has been made for (a), input request for (a) should NOT be repeated).

4. All calculated numbers (including the check-sum) should be displayed correctly (no extra number(s), no missing number(s), and no any unnecessary output) as shown in the figure (note that there is no comma after the last number).

5. After all the numbers are generated and displayed, their “check sum” should be displayed (the correct check sum should be displayed). The check-sum is a total sum of all the numbers generated in a number sequence.

(3) SWICTH ~ CASE ~ structure

This program first prompts a user to enter two numbers: this year (2022 - 2122) 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 borne by:
The year you were borne = (this year) – (your age)

Step 2: Subtract 1900 from the year you were borne. 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 3:00 p.m. on September 23rd.

- Extra-Credit (+3 points to your midterm exam score) Early-Submission³ Deadline: 11:59:59 P.M., September 18th
- Free-Feedback Early-Submission³ Deadline: 11:59:59 P.M., September 20th
- Final Deadline: 3:00 p.m., September 23rd

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 CS286-002 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 12 hours after the due (i.e., -10% for a submission within the first 12 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. Dr. Fujinoki will not debug your programs. Please do not e-mail your *.asm source code files to Dr. Fujinoki unless you are requested to do so.
2. Identify where (in your source code) the problem exists.
3. Describe the symptom(s) of the problem.
4. Describe how the problem happens (always happen, sometime happen, the condition(s) for the problem to happen, etc.).
5. Describe what you tried (to understand and/or solve the problem).
6. Stop by Dr. Fujinoki’s office (no question through e-mail).