Summary: Many applications that process large data sets use sparse data representations for storage and processing efficiency. This assignment explores the use of sparse vectors, basic conditional execution, nested iteration, and memory access operations in analyzing time/space data, such as information collected by GPS-enabled applications. In particular, you will write a program that analyzes two timelines, each of which gives a history of major cities in which a person has lived in chronological order. Your program will determine the earliest year in which both people lived in the same city.

Data in each timeline TL is stored as a sparse vector of ten elements in the following format (for i=0, 1, …, 4):

```
TL[i*2] = Duration: number of years a person lived in the city
          that is specified in TL[i*2+1].
TL[i*2+1] = City in which the person lived for TL[i*2] years.
```

Objective: For this assignment, you will write two programs, one in C and one in MIPS, to compute the earliest year at which Harry and Sally both lived in the same city. More details are described below.

Assume that the two people were both born in the year 1990 and the current year is 2019. So the total number of years (sum of TL[i*2] for i=0,1,…,4) is always 30. Each city is an integer between 0 and 9, inclusive. Also, assume that the total number of moves in each timeline is exactly five. For example, the timelines shown in Figure 1 are represented in C as:

```
HarryTimeline[] = {4, 4, 9, 3, 3, 8, 2, 4, 12, 2};
SallyTimeline[] = {1, 3, 11, 2, 4, 4, 6, 2, 8, 4};
```

Miami has city code 4 and Harry spent 4 years living there, then moved to Atlanta (city code 3), where he lived for 9 years. For this example, your program should compute 2008 as the correct answer.

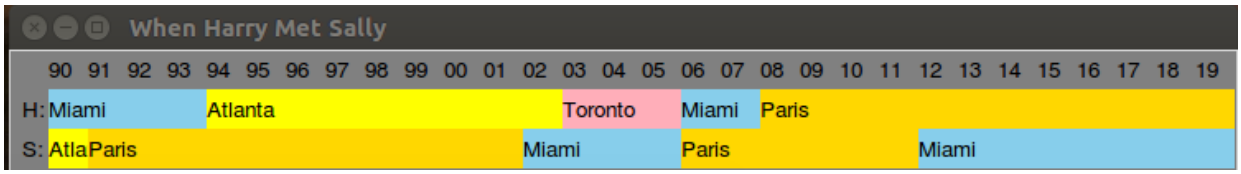

Figure 1: Timelines showing how long Harry and Sally lived in certain cites. The earliest year in which they both lived in the same city is 2008 (when Harry moved to Paris where Sally was already living).

Note that this sparse vector representation is much more storage efficient than storing the city for each time point in the range 1990-2019.

For this assignment, you will write two programs, one in C (part HW2-1) and one in MIPS (HW2-2).

HW2-1: For this part, design and implement a C program to compute the earliest year at which Harry and Sally both lived in the same city. As a starting point, use the file `HW2-1-shell.c` and print the year computed using the print string provided in the shell code. If there is no year in which they are both in the same city, print 0. Name the file containing your solution `HW2-1.c` and upload it to Canvas by the scheduled due date.

The provided shell program includes a reader function `Load_Mem()` that loads the values from a text file. Sample test files (`test1990.txt, test0.txt, test2004.txt, test2008.txt` in `tests.zip`) are provided – the number in the name of each file indicates the correct answer. Unzip the file into a subfolder named `tests` under the directory containing your `HW2-1.c` code.

```
[gburdell@ece-linlabsrv01 hw2]$ unzip tests.zip
```

You should compile and run your program using the Linux command lines (this is what the autograder will use):

```
[gburdell@ece-linlabsrv01 hw2]$ gcc HW2-1.c -g -Wall -o HW2-1
[gburdell@ece-linlabsrv01 hw2]$ ./HW2-1 tests/test1990.txt
```

(Replace "test1990.txt" with other test file names to run more tests.)

We have also provided a short "**smoke test" script (run_tests.sh)** with the HW2-1 Assignment on Canvas. *Please place* **run_tests.sh** *in the same directory where you put your HW2-1.c code.*

You must run the smoke test on your code on the Linux Lab servers before submitting your code and fix any errors detected.

```
[gburdell@ece-linlabsrv01 hw2]$ ./run_tests.sh
```

If this gives the error "bash: ./run_tests.sh: Permission denied" be sure to change the permissions on the bash script file (run "man chmod" at the command line for more information):

```
[gburdell@ece-linlabsrv01 hw2]$ chmod u+x run_tests.sh
```

The **run_tests.sh** script will point out errors that you should fix before submitting your code. The output will also be saved to a text file **log.out** in the same directory.

Be sure to run several more tests. The smoke tests do not represent the comprehensive set run by the autograder. You can create additional test files for your C program using MiSaSiM in this way: run `HW2-2-shell.asm`, go to the end of the trace, and use the "Dump" memory menu button to save the memory to a text file with the correct answer (the value in $3, as described below) in the name of the file.

Make sure that you understand the usage of the `DEBUG` constant, as described in the comments of the shell. You will need to set `DEBUG` to 0 to suppress any debug messages that you may have included, both for your own output comparisons and for actual grading.

In order for your solution to be properly received and graded, there are a few requirements.

1. The file must be named **`HW2-1.c`**. (It is OK if Canvas renames it slightly with an extended version number (e.g., `HW2-1-2.c`; we'll grade your most recent submission.)
2. Your name and the date should be inserted in the header comment.
3. *Do not* `#include` *any additional libraries.*
4. Before submitting your code, reset `DEBUG` to 0 and run your code through the smoke tests on the Linux Lab servers to ensure that your code is providing the answer in the proper format without any extraneous print statements. ***If your submitted code has a bug that the smoke test would detect, you will receive a 0.***
5. Your solution must include proper documentation (comments) and appropriate indentation.
6. Your solution must be properly uploaded to Canvas before the scheduled due date.

HW2-2: Design and implement a MIPS version of your program. A description of the MIPS library routines you need is given below. Use the file `HW2-2-shell.asm` as a starting point. The result should be stored by your program in **$2** and reported as an answer by software interrupt **swi 587** as described below.

You must use these register conventions or the automated grader will score your program incorrectly. **Memory should only be used for input to your program. (Only registers should be used to hold intermediate and final results.)**

MIPS Library Routine: There are two library routines (accessible via the `swi` instruction).

SWI 597: Create Duration Timelines: This routine generates and displays the timelines as shown in the example in Figure 1. It initializes memory with the 20 integers beginning at the specified base address (e.g., `Harry`).

> INPUTS: $1 should contain the base address of the 20 words (80 bytes total) already allocated in memory.
>
> OUTPUTS: the 20 words allocated in memory contain the 20 sparse vector elements (alternating duration and city) to be used as input data.

SWI 587: Report Earliest Overlap: This routine allows you to report your answer and an oracle provides the correct answer so that you can validate your code during testing.

> INPUTS: **$2** should contain the year you computed as your answer.
>
> OUTPUTS: **$3** gives the correct answer.

If you call **swi 587** more than once in your code, only the first answer that you provide will be recorded.

Easter egg: As a debugging feature, you can load in a previously dumped testcase (pair of timelines) by putting -1 into register $2 before you call swi 597. This will tell swi 597 to prompt for an input txt file that contains a testcase (e.g., `test1990.txt`).

For your solution to be received and graded, there are a few requirements.

1. Your file must be named **`HW2-2.asm`.** (As mentioned above, it is OK if Canvas renames it slightly with an extended version number; we'll grade your most recent submission.)
2. Your name and the date should be inserted at the beginning of the file.
3. Your program should store the result in **$2** and call **swi 587** to report it.
4. Your program must return to the operating system via the **`jr`** instruction.
5. *Programs that include infinite loops or produce simulator warnings or errors will receive zero credit*.
6. Your solution must include proper documentation.
7. Your solution must be properly uploaded to Canvas by the scheduled due date.

Honor Policy: In all programming assignments, you should design, implement, and test your own code. **Any submitted assignment containing non-shell code that is not fully created and debugged by the student constitutes academic misconduct.** The only exception to this is that you may use code provided in tutorial-0.zip or in the examples on the course Canvas site as a starting point for your programs (Canvas>Modules>_Module_Name_>Code Examples).

All code (MIPS and C) must be documented for full credit.

Happy coding!