# CSE 465/565
# Homework #6
# Spring 2019

Notes:

- You will have 3 separate submissions for this assignment:
    - HW6 Reports - a PDF file that contains your two reports
    - HW6 parse.py - `parse.py`
    - HW6 Timing Code - zip file containing `demo.{cpp,cs,py}`
        - You will probably modify these files several times to get all the timings needed to write your report. You do not need to submit your code for each of these intermediate stages. Instead, submit the code used to obtain the timings you used to answer the first task listed in the table below. You will have three code files, one for each languages.
- Your code will be tested on `ceclnx01`. Any code that does not work on that machine will receive a score of 0. The different languages will be run using the following commands, where "`demo`" is the basename of the file:
    - `python3 demo.py`
    - `mcs -optimize- demo.cs`
      `mono demo.exe`
    - `c++ -O0 demo.cpp`
    - `javac demo.java`
      `java -Djava.compiler=NONE demo`
- Your code must have the correct spelling for filenames and function names. You may add code to the file but do not change the spelling, case, number of parameters, etc. Code that does not work with the tester will receive a score of 0.
- Use the following command to get a copy of the starter code:
    - `$ cp -r ~zmudam/CSE465-565/HW6S19/ .`
- An example of my test script for problem #1 is provided and is named `testParse.py`

**1. (50/40 pts)** Write a Python boolean function, `sentence`, defined in a file named `parse.py` to detect legal English sentences, as defined by the vocabulary/rules specified in `parse.pl`, which was distributed to you earlier this semester.

The input to your function will be one string consisting of lowercase, space-separated, words. Your function will be called in a manner similar to the following:

```
if sentence('the yellow sun shines'):
      print('Legal')
else:
      print('Illegal')
```

Your code does not have the use the same problem-solving strategy used in the Prolog code; you can solve the problem in any way you wish. A portion of your score will be based on programming style (see below).

Write a report that compares your new code with the original Prolog code. Include any aspect that you think is noteworthy/interesting to a fellow CSE 465/565 student. For example, is either solution more readable than the other? Would one solution be more maintainable than the other? etc. The body of your text (i.e., excluding any code samples) should be less than 1 double spaced page.

**Scoring.**

- 18/8 pts. Correctness of your code
- 20/20 pts. Report
- 12/12 pts. Program style. Your code should be highly polished and utilize good programming practices. It should be easy to introduce new nouns, verbs, and articles. In addition, your code should use appropriate data structures and Python features at every reasonable opportunity.

**2. (0/10 pts) CSE 565 students only.** Solve the previous problem again (in a function named `sentence2`) but using the sentence definition defined in problem #5 from Homework #3 (i.e., the plurality of the subject and verb must agree). In addition, adjectives cannot be duplicated in the same sentence. You do not have to rewrite the report.

**3. (50 pts)** In this problem, you will investigate the time taken to pass parameters (possibly large data structures) to functions. You will examine different languages, different parameter passing modes, and differing ways that a data structure can allocate its data. Files `demo.{cpp,cs,py,java}` have structures and sample functions that you are to use. You will need to modify this code to get the timing data needed for your analysis.

When doing timings, it is important to allow the program to run long enough to obtain meaningful times. Running a program on a small data set may execute very quickly and give inconclusive results. For example, you might get timings of 0.1s and 0.08s. These values are small and somewhat similar. In this situation, it is difficult to determine if the time differences are the result of the algorithm or simply startup overhead. It is important to minimize the impact of any initial overhead and other anomalies. To do this, you should increase the data size, or increase the number of repetitions, so that the execution time for the specific experiment is significant: > 3s, for example. Then, using the same number of iterations/data size, execute the second algorithm and record its timing. For the sake of example, let's assume that the second algorithm takes 8s. You could present this as: “Algorithm 2 took 60% more time than Algorithm 1.” At this point, you would then need to explain why Algorithm #2 is significantly slower than the first. Be concise but include enough detail so that a fellow CSE 465/565 student can understand your rationale. One example has been done for you in the table below.

We want to run programs without any optimizations (so that we can see only the effects of the different parameter passing approaches). That is what `-optimize-` does for `mcs` and `-O0` does for `c++` (see instructions at the top of this writeup).

Lastly, if you are doing your timings on a system with many users, such as `ceclnx01`, you should get your related timings at the same time. This will help to ensure that the other users’ activities are affecting your program’s speed consistently. It is also a good idea to run them a couple times to get an average time.

Your report will consist of the following completed table (the first one has been done for you): SEE NEXT PAGE.

| **Lang.** | **Question Regarding Time to Perform Parameter-Passing** | **Summary** | **Explanation** |
|---|---|---|---|
| Java | What effect does array length have when calling function `f`? | Call times are are nearly identical, regardless of array length | The reason for this is that Java arrays are passed by reference. That is, the array itself is allocated on the heap and only a pointer to the array is sent to the function; the array itself is never copied onto the stack. The number of bytes in this pointer is the same for large or small arrays. Thus, the overhead of each function call is the same. |
| Python | The class `MyStruct` contains a list. What is the effect of the length of this list when calling the function `f`? | | |
| C# | The `MyStruct` contains 10 `doubles`. What is the difference when passing this structure as a value parameter vs. a `ref` parameter? | | |
| | What is the effect if the number of `doubles` inside the struct is switched from 10 to 20? | | |
| | What effect does defining `MyStruct` as a `class` instead of `struct`? | | |
| C++ | For `StructF`, what is the effect of the array length? Consider all three parameter passing approaches. | | |
| | For `StructG`, what is the effect of the array length? Consider all three parameter passing approaches. | | |
| | For pass by value, how do `StructF` and `StructG` compare, given identically sized arrays? | | |
| | For pass by reference, how do `StructF` and `StructG` compare, given identically sized arrays? | | |
| | What is the effect of defining `StructF` as a `class` instead of `struct`? Does this match with C#? | | |