#3 CS/SE 2340 Assignment
FALL SEMESTER 2025, Sections 002, 004, 006, 501

INSTRUCTOR: DR. ALICE WANG

Submission (ZIP file):

- Assembly code (.ASM) file. Use and Include SysCall.asm in your ZIP file.
- PDF showing screenshots of your code working

Submit all of your work in a ZIP file to eLearning by the due date. If you are not familiar with the ZIP file format you can find out more about it from this link. IMPORTANT: do not use another archive format, e.g. RAR, because the grader will not be able to see your files, and you will get 0 points.

Note: name your ZIP files for homework submission as follows:

HW<hw#>_<FirstName>_<LastName>.ZIP, e.g. HW03_Jane_Doe.ZIP

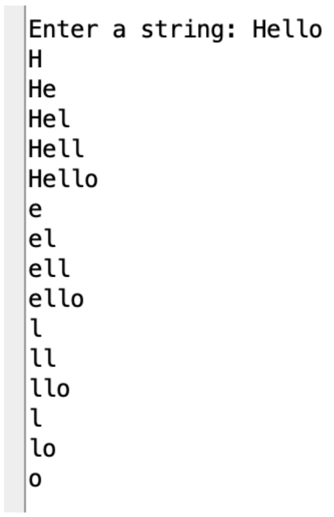
For assignments of this class you can submit your work unlimited times, the last submission will be graded.

Goal: Write a MIPS assembly program that performs dynamic memory allocation, string manipulation, and substring generation using system calls and loops.

Part 1 - String manipulation

1. Heap Allocation:
   Dynamically allocate 1024 bytes of memory on the heap using the SysAlloc system call (service code 9 in SysCalls.asm). This is similar to the malloc() function in C.
2. User Input:
   Prompt the user to enter a string. Store the input string in the heap-allocated memory.
3. String Length Calculation:
   Use a for loop to determine the length of the input string.
4. Substring Generation:
   Using nested for loops, generate and print all possible substrings of the input string. Substrings should include spaces and be printed in the order they appear.
5. Reference Material:
   Refer to the provided pseudocode for guidance on the substring generation algorithm

using iteration. See below for an example run for debugging, but ensure your final submission includes 2 different input strings.



```
Enter a string: Hello
H
He
Hel
Hell
Hello
e
el
ell
ello
l
ll
llo
l
lo
o
```

Testing and Submission

- Assemble and run the program using MARS
- Capture screenshots of two different runs showing input and output
- Submit a ZIP file containing:
    - Your .asm file
    - SysCalls.asm
    - A PDF report with screenshots
- Ensure your .asm file includes:
    - Your name and header comments
    - Proper use of SysCalls.asm
    - Full commenting on every line
    - A clean program exit using the exit syscall

Part 2 - CPU performance

Using the "Instruction Count" tool in MARS, calculate the instruction count for the substring parsing code for the following input strings.

String1 = Life is like a box of chocolate

String2 = Why chase your dreams when you can nap and let them come to you with snacks and a cozy blanket?

Set breakpoints while executing your code before the loop portions of code and note down the difference in number of instructions before and after the loops executes. Calculate the Execution time for the for-loop for these two strings, assuming a 2GHz clock and CPI = 1.

Explain if this code is linear time `(O(n))`, quadratic time `(O(n²))` or cubic time `(O(n³))` and why.

Assignment Grading Rubric

Criteria	Points Deducted
Missing .asm extension	-5 pts
Missing or incorrect syscall usage SysCalls.asm not included in ZIP	-5 pts
Missing header or author info	-5 pts
Poor or missing comments on every line	-15 pts
Program does not exit properly	-5 pts
Program does not compile	-30 pts
Missing 2 screenshots	-15 pts each example
Did not dynamically allocate to the heap	-10pts
Did not use for-loops	-20pts
Does not correctly print the substrings	-10pts
Did not report on the Instruction Count and the program execution time for each string	-5pts for instruction count per string -5pts for execution time per string