**Project Description:**

Write and submit a NASM assembly language program “shift.asm” that:

* Asks the user for a number between 1 and 99. This number will be used as the seed for the random number generator. The random number generator **must be** implemented by converting this C code below into x86-64 assembly.

int maxrand(int seed,int max)

{

int random\_seed=0;

random\_seed = random\_seed+seed \* 1103515245 +12345;

return (unsigned int)(random\_seed / 65536) % (max+1);

}

* Asks the user for a string from the user and displays the unedited string.
* And finally performs a transformation on the string from either the reverseHalves or scramble operations and displays it. *The selection of which operation is applied must be done randomly.* Transforming the string should not change the original string. Here is the explanations for each of these transformations:

* reverseHalves: The idea is to divide the input text into two halves, then reverse each half and finally put these halves back into one string. For instance, if the input text is “An apple a day keeps the doctor away!”, after applying reverseHalves to it the result would be: “eek yad a elppa nA!yawa rotcod eht sp”. reverseHalves does not change empty, single character, and two-characters-long texts.

* scramble: This one “scrambles” the text by putting first and last characters next to each other, then it puts the second and the second last characters next to each other, then the third and third last, so on so forth all the way to the middle point of the text. As an example, if the text is “An apple a day keeps the doctor away!”, after scrambling it we would have “A!ny aawpap lreo tac odda ye hkte esp”. scramble does not change empty or single character texts.

**General Project (Hard) Requirements:**

* **Code that does not assemble will receive 50% off**
* **We are programming for 64 bit Intel Architecture**
  + **Code written for 32 bit Intel will receive 20 points off**
* **You can work individually or with another person for this project. No more than 2 people can work together.**
* You must have a comment at the top of the code detailing what the code does.
* This comment should also include your full name and user ID.
  + If working with 2 people both names and user IDs should be included.
* You must use good coding style with respect to variable names, spacing, labels and comments.
* You cannot use C/C++ function calls. **You have to use system calls**

**Grading Breakdown: (remember that code that doesn’t compile will not get full credit)**

* [75] Functionality
  + [20]Display prompts to user and read in values
  + [10]Display unedited string
  + [10]Display edited string
  + [20]Text edit functionality
    - Including random number generation
  + [5]Use of system calls
  + [5]Program exits correctly (No Seg Faults)
  + [5]No extra new lines or spaces printed
* [25] Style (only if it compiles)
  + [5]Comments where necessary
  + [5]Code written for 64 bit Intel Assembly
  + [5]Comment at top of program with name, user id and explanation of code
  + [5]Code easy to read
    - Good variable names
    - Indentation
  + [5]Submitted correctly