Assignment 4: Hangman Game

NYU's Intro to Computer Science
Section # 6 and 9
Professor Sana Odeh

Due on 11/13 by Midnight!

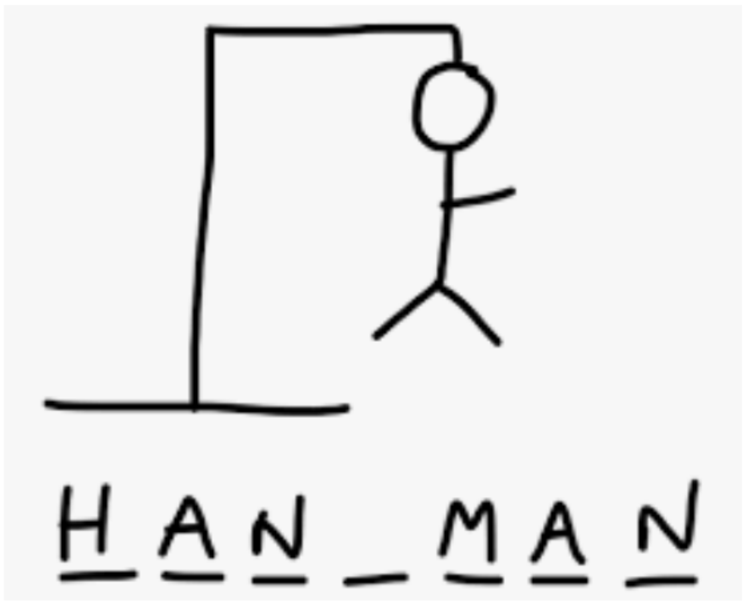
# Design a Hangman game! You must use arrays, methods and loops!

## 1 Code of Conduct

All assignments are graded, meaning we expect you to adhere to the academic integrity standards of NYU. To avoid any confusion regarding this, we will briefly state what is and isn't allowed when working on an assignment.
Any document and program code that you submit must be fully written by yourself. You can, of course, discuss your work with fellow students, as long as these discussions are restricted to general solution techniques. Put differently, these discussions should not be about concrete code you are writing, nor about specific results you wish to submit. When discussing an assignment with others, this should never lead to you possessing the complete or partial solution of others, regardless of whether the solution is in paper or digital form, and independent of who made the solution. That means, you are also not allowed to possess solutions by someone from a different year or course, by someone from another university, or code from the Internet, etc. This also implies that there is never a valid reason to share your code with fellow students, and that there is no valid reason to publish your code online in any form.
Every student is responsible for the work they submit. If there is any doubt during the grading about whether a student created the assignment themselves (e.g. if the solution matches that of others), we reserve the option to let the student explain why this is the case. In case doubts remain, or we decide to directly escalate the issue, the suspected violations will be reported to the academic administration according to the policies of NYU (see https://cs.nyu.edu/home/undergrad/policy.html).

# Hangman Game Application:

## Part A: Design a Hangman game using arrays, methods and loops!



Hangman is a simple word guessing game. Players try to figure out an secret word by guessing letters. If too many letters which do not appear in the word are guessed, the player is loses the game.

Setup the game by drawing a gallow (see above photo or you can use any other figure, not necessarily a hanged man). Provide an underline (underscore character (_ ) for each letter in the secret word. As letters in the word are guessed, write them in the exact position in the word replacing the underscore character with the guessed correct letter. If a letter is not in the word, draw a picture of a person on the gallow–one part for each incorrect letter guess. Most frequently, the person is drawn in 6 parts (for 6 letter guesses) in the order: head, body, left leg, right leg, left arm, right arm.

**Here is more information:**

- Create an array with 10 words (animals' names).
- Words should be selected randomly (using a random method).
- Make sure that when the player guesses the correct letter, your program should output the correct letter in the exact position, otherwise, it should keep the "_" in the same position.

- For example, if the secret word is "cat", then you need to display underscore character "_" equal to length of the secret word. In this case it will display three underscore characters since the length of the string is 3 "___" dashes.
- When the user guesses the correct letter, then you have to display the letter in the correct position and also display all of the occurrences of that letter in the string. For example, if the user guessed the letter "a", then you should display the letter "a" in the correct position printing "_a_". Make sure to print the string after each guess.
- If the user makes the wrong guess by typing "o" for example, then you should display the string with no changes such as "_a_" and you should **remove a piece from the stick figure or add a piece as you wish.**
- **Use characters to draw a stickperson needed for the game such as "_", "O", "|" and make sure to remove one character/piece from stickperson with each wrong guess. Feel free to use other characters to draw a creative stickperson or change the figure entirely to make it more politically correct.**
- The number of guesses allowed for each game is based on the number of characters in the **stick figure (usually it's 6: two legs, two hands, body, and a face). With each wrong guess, you remove one character/piece from the stick figure. When all of the 6 characters are removed from the stick figure, then the user loses the game.** Your program should inform the user that she had won or lost the game.
- The player should win right away when they guess all of the letters correctly within 6 trials (there should be at least one piece in the stick figure remaining for you to win the game).
- If the user lost the game, your program should print the "secret" word.
- Feel free to change the number of trials and adding an easy, medium difficulty and a hard level for this game for extra credit.
- Make sure to write your own code and NOT to copy any line(s) from the web, classmates or any other resources! You will get a zero.

**Extra credit will be provided for excellent design, creativity and added technical features.**

**Part B:**
Redesign the above hangman/snowperson application from above, but now you need to get the data (the words) from a file on your computer called animals.txt. The program should read the words one line at time from the file and then assign the words to an array. Then follow the same code as you did for part1.

**Part C:**

Redesign the same hangman application from above, but now you need to scrape /get the words from a file on the web from this link (https://cs.nyu.edu/~odeh/resources/python/animals.txt).

Submit all .java and animals.txt files (make sure to include the .txt) when done through brightspace.

Please let us know if you need help as we are happy to help.

**Grading**

| **Description** | **Points (/100)** |
|---|---|
| Array and guessing the word randomly | 10 |
| Game strategy- guessing the word in 6 trials | 35 |
| Stick figure manipulation during play (removing parts) | 10 |
| Outputting the string with each trial | 5 |
| Outputting the result (if games lost, or won and outputting the secret word when game is lost) | 5 |
| Reading words from file on your computer animal.txt (Use try and catch) | 15 |
| Reading files from the web | 15 |
| Code aesthetics (comments, readability of code, style, variable naming, etc.) | 5 |

**Extra credit will be provided for excellent design, creativity and added technical features.**

**Submission**

The deadline of this assignment is (11/13 by midnight) after its release via Brightspace. Place all the files needed for this assignment in a .zip folder and submit it to Brightspace. Submissions via email are not accepted. Late submissions will be penalized by 5% per 24 hours and **Late homework will not be accepted after 3 days from the due date**.
Note that your solution must work using java with no syntax errors. In case your code does not work using Java, your submission will not be graded.