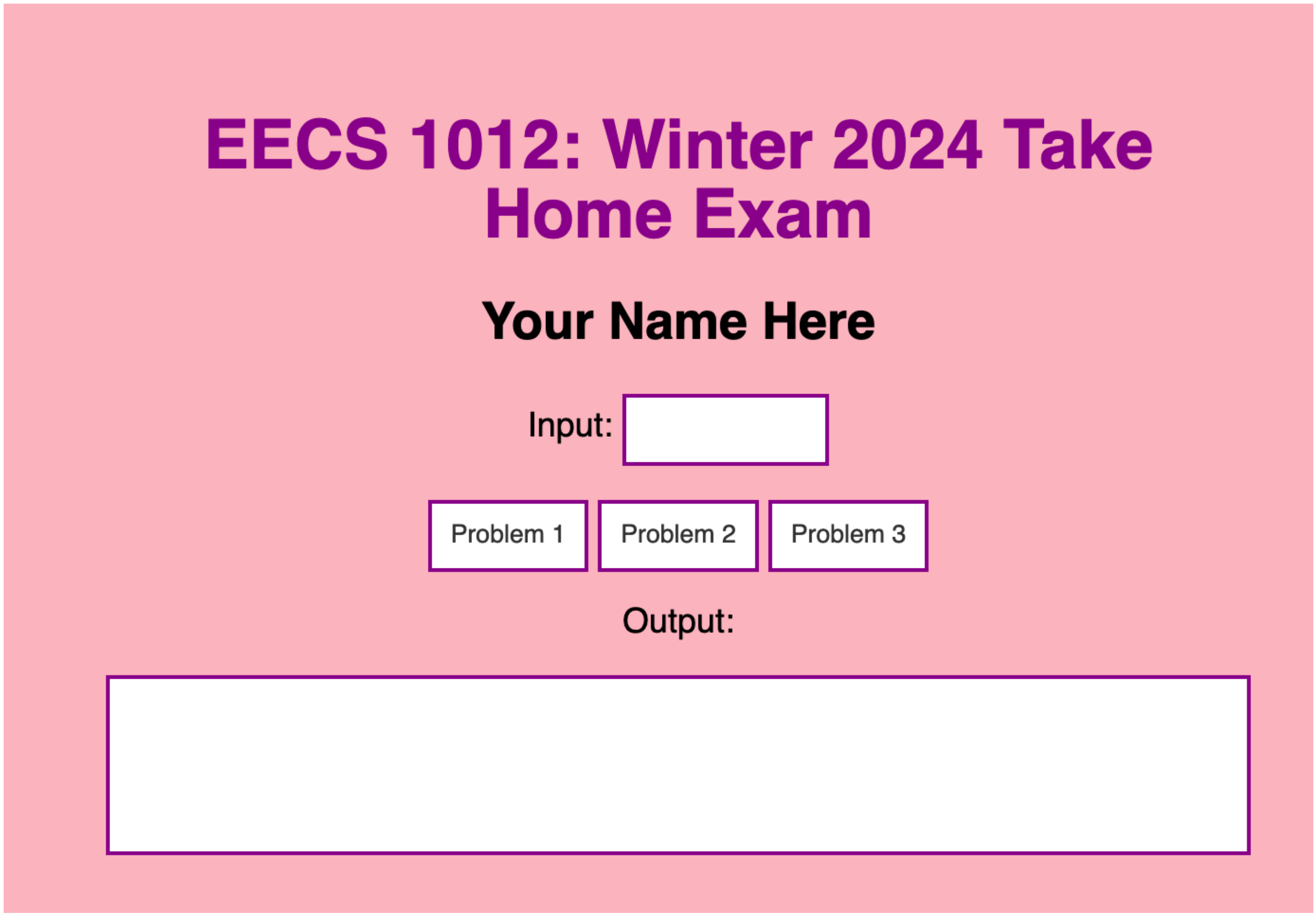
Important notes:

1) You must work alone on this take home exam. You may ask clarification questions to the instructional team, but you cannot ask for instructors to review your solutions. You may reference code and consult https://www.w3schools.com/, https://validator.w3.org/, and https://jigsaw.w3.org/css-validator/ as needed.
2) **No inline or embedded styling or scripting** is allowed inside your `HTML` file.
3) Make sure to provide consistent indentations in all your coding files.
4) **eClass may be used to submit your work.**

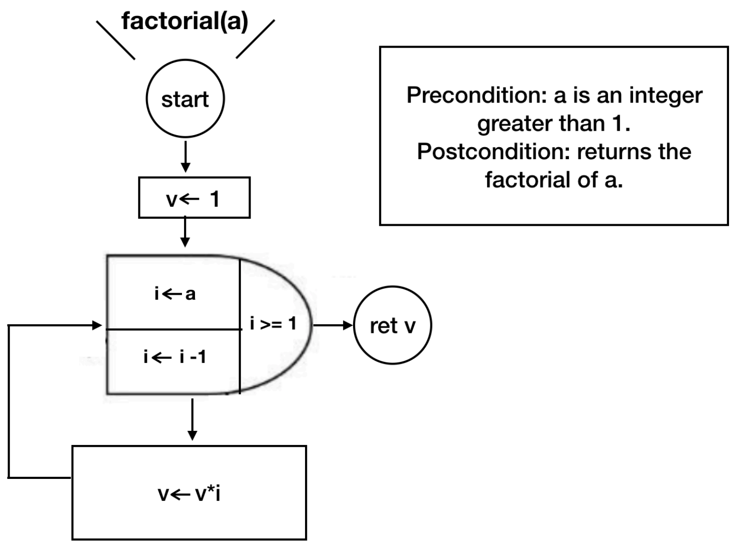
Rename takehome.html, takehome.css and takehome.js as follows: call the `HTML` file **`<yourfullname>.html`,** the css file **`<yourfullname>.css`** and the JS file **`<yourfullname>.js`**. Replace `<yourfullname>` with your full name **without any spaces.**

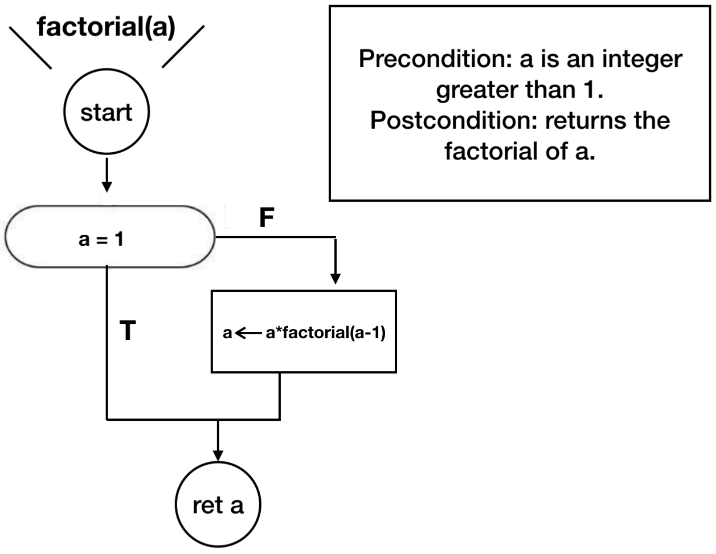


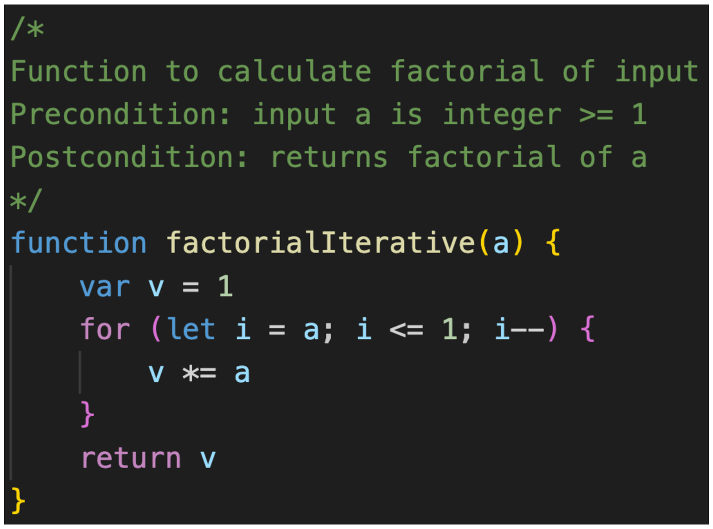
Step 1 [worth 60 points]: Craft your `HTML and CSS` files so that they realize a page that looks like the example on page 1. You should:

- **A)** include your name within a comment in both files and indicate you are the author (worth 6 points);
- **B)** in your HTML file, include an **h1** header for the title, an **h2** header for your full name, an **input box or area**, an **output box or area**, and **three buttons** with labels as shown in the figure (worth 10 points);
- **C)** Ensure that each button triggers a different method in your JS file **upon a click**. These methods should be named: **`program1(), program2() and program3()`** (worth 12 points)
- **D)** in your CSS file, add a page background that is **pink**, as well as an input area, output area and buttons that are **white**. You should include a **border** on the input area, output area and buttons; the dimensions of buttons and boxes, as well as the colour and thickness of borders can be of your choosing. You may add other styling as you see fit; this additional styling will not be marked for correctness, however. (worth 12 points);
- **E)** connect your HTML to your CSS and JS file (worth 10 points);
- **F)** ensure your HTML and CSS are valid (use https://validator.w3.org/ to validate HTML and https://jigsaw.w3.org/css-validator/ to validate CSS) (worth 10 points).

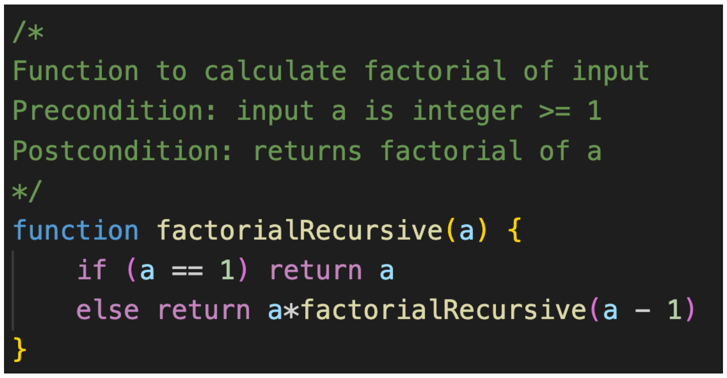
Step 2 [worth 40 points]: As we have discussed in class, any recursive function can be written as an iterative function. Below are flowcharts that illustrate two implementations of the "factorial" method; the one at the right is **recursive** and the one at the left is **iterative**. On the next page you will find the javascript implementations of either method. The implementation to the right is the **recursive** and to the left the implementation is **iterative**.


factorial(a)
start
v← 1
i←a
i← i -1
i >= 1
ret v
v← v*i
Precondition: a is an integer greater than 1. Postcondition: returns the factorial of a.


factorial(a)
start
a = 1
F
T
a← a*factorial(a-1)
ret a
Precondition: a is an integer greater than 1. Postcondition: returns the factorial of a.



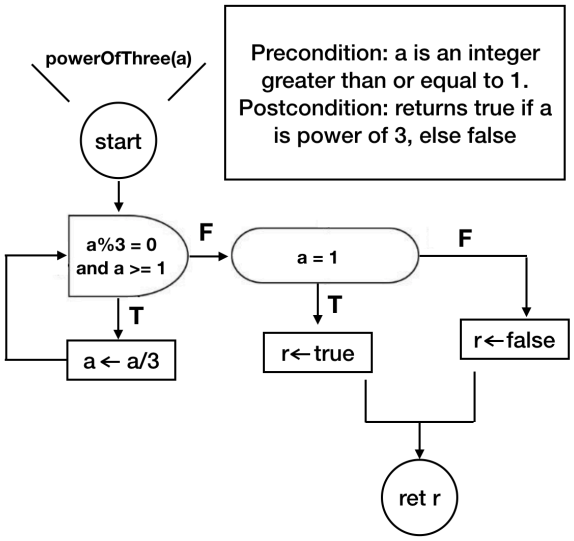
```
/*
Function to calculate factorial of input
Precondition: input a is integer >= 1
Postcondition: returns factorial of a
*/
function factorialIterative(a) {
    var v = 1
    for (let i = a; i <= 1; i--) {
        v *= a
    }
    return v
}
```



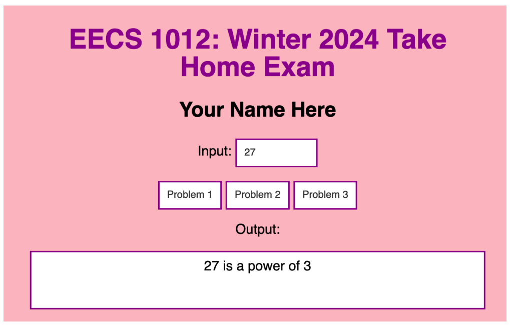
```
/*
Function to calculate factorial of input
Precondition: input a is integer >= 1
Postcondition: returns factorial of a
*/
function factorialRecursive(a) {
    if (a == 1) return a
    else return a*factorialRecursive(a - 1)
}
```

In the questions that follow, we have provided flowcharts containing **iterative** solutions to a problem. Your job is to translate these **iterative** solutions to **recursive** solutions. However, you will receive half marks if you implement the **iterative** solutions that are illustrated in the flowcharts. Note that each problem is independent of the others. Also note that our flowcharts do not handle invalid (i.e. non-numeric) inputs, but we ask that your implementations handle such cases. You can develop your functions in any order you choose.

A. **[Worth 13 points] Function `program1()`**. When button with the label "problem 1" is clicked, the function **`program1()`** should be called in your JS file. This method should call a **recursive helper function** with logic equivalent to that in the **iterative flowchart** that is shown below. This **recursive** function (and its iterative analogue) determines if a user's input in the input box is a power of three. If the user enters **3, 27, or 9** in the input box, the function should indicate that the input IS a power of three. If the user enters **6 or an alphabetical character**, the function should indicate the input IS NOT a power of three. An example of what your program should look like when different inputs are provided to your HTML form is illustrated below.



powerOfThree(a)
Precondition: a is an integer greater than or equal to 1. Postcondition: returns true if a is power of 3, else false
start
a%3 = 0 and a >= 1
F
T
a ← a/3
a = 1
F
T
r←true
r←false
ret r



EECS 1012: Winter 2024 Take Home Exam

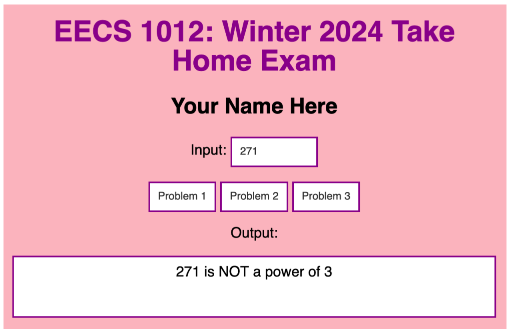
Your Name Here

Input: 27

Problem 1 Problem 2 Problem 3

Output:

27 is a power of 3



EECS 1012: Winter 2024 Take Home Exam

Your Name Here

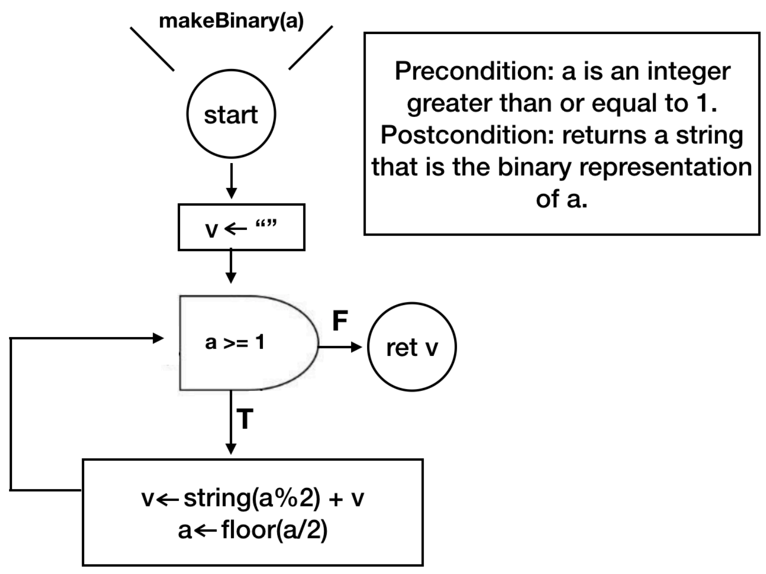
Input: 271

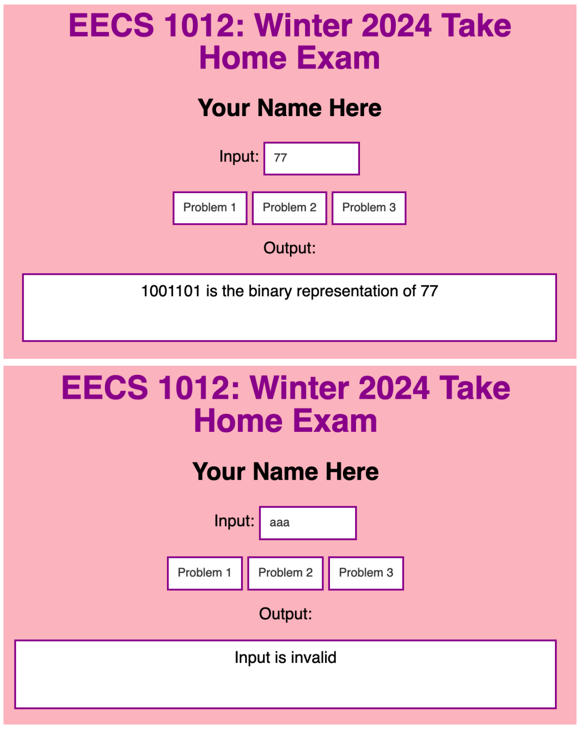
Problem 1 Problem 2 Problem 3

Output:

271 is NOT a power of 3

B. **[Worth 13 points] Function `program2()`.** When button with the label "problem 2" is clicked, the function **`program2()`** should be called in your JS file. You will write a **recursive helper function** for this function with logic equivalent to the **iterative** flowchart below. This **recursive** function (and its iterative analogue) converts a decimal integer into a binary string (as does https://www.binaryhexconverter.com/decimal-to-binary-converter). If the user enters **9** in the input box, the function should output the string "1001". If the user enters **22**, the function should output the string "00010110". However, if the user enters an **alphabetical character**, the function should output "Invalid Input". Our flowchart assumes two sub-algorithms: floor and string.


makeBinary(a)
start
v ← ""
a >= 1
F
ret v
T
v← string(a%2) + v
a← floor(a/2)
Precondition: a is an integer greater than or equal to 1. Postcondition: returns a string that is the binary representation of a.



EECS 1012: Winter 2024 Take Home Exam

Your Name Here

Input: 77

Problem 1 Problem 2 Problem 3

Output:

1001101 is the binary representation of 77

EECS 1012: Winter 2024 Take Home Exam

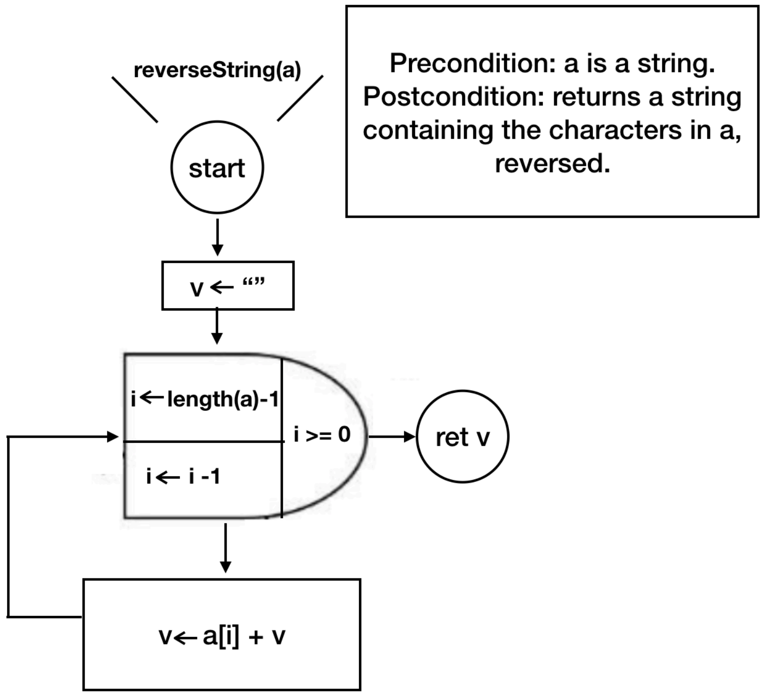
Your Name Here

Input: aaa

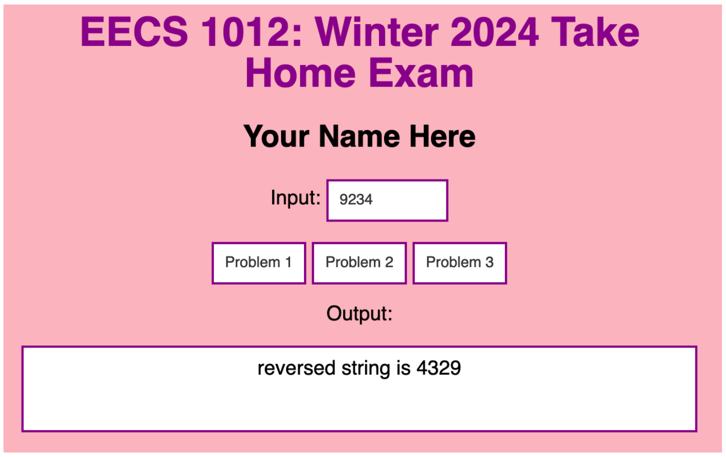
Problem 1 Problem 2 Problem 3

Output:

Input is invalid


reverseString(a)
start
v ← ""
i←length(a)-1
i >= 0
i← i -1
ret v
v←a[i] + v
Precondition: a is a string. Postcondition: returns a string containing the characters in a, reversed.

C.**[13 points] Function `program3()`.** When button with the label "problem 3" is clicked, the function **`program3()`** should be called in your JS file. You will write a **recursive helper function** for this function with logic equivalent to the **iterative** flowchart to the left. This **recursive** function (and its iterative analogue) reverses the characters in a string. If the user enters 9234, the function should output the string "4329". If the user enters "superduper", the function should output "repudrepus". Your function should be able to handle both numeric and non-numeric input. Some examples of what your HTML should look like can be found on the next page. Note also our flowchart assumes a sub-algorithm to determine input length.

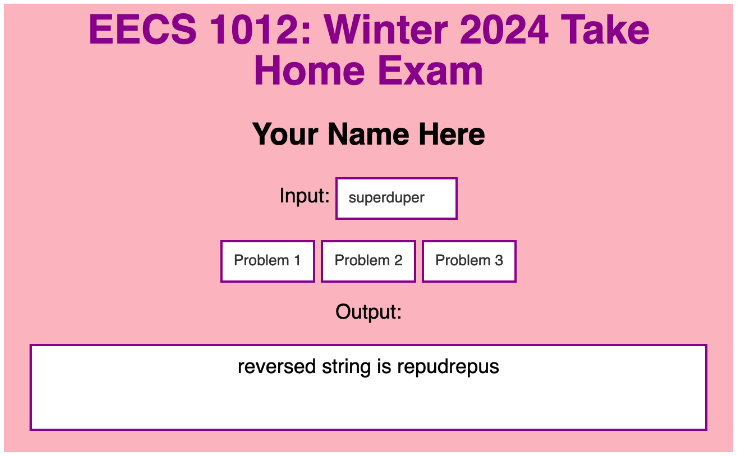

EECS 1012: Winter 2024 Take Home Exam

Your Name Here

Input: 9234

Output:

```
reversed string is 4329
```


EECS 1012: Winter 2024 Take Home Exam

Your Name Here

Input: superduper

Output:

```
reversed string is repudrepus
```

D. **[Worth 1 point]** Finally, to complete the exam, add your name to the text file named "acknowledgment.txt". Submit this file along with your CSS, HTML and JS. **You may submit your work individual files or as a zipped archive. Submit to the exam page on eClass anytime before Tuesday night (4/2), 9 pm.**