WEEK 6 REPORT

(___/1pt) (y/n) your product is effective to deliver the knowledge in computer science to the product users.

- n - The code might help deliver the knowledge, but the product is more related to English Language since it is a word guessing game. (English Vocabulary)

(___/1pt) user friendly/appealing in terms of the following criteria.
(y/n) The landing page is attractive. (hints: the homepages of the hightech giants)

- y

(y/n) Users are be able to understand and play the puzzle game quickly.

- y

(y/n) Users can just jump in and start playing (trying out) the game immediately without the registration process.

- y

(Certainly, the performance progress of unregistered users will not be recorded.)
(___/1pt) Your product should have the following functions. .
(y/n) Users can register with a username and a password.

- y

(y/n) The performance of registered users are updated after each trial and can be displayed upon requests

- y

(y/n) Users can ask for hints and/or solutions.

- My product is a puzzle solver (A solution to the problem, so it doesn't have any hints)

(y/n) Administration account

- Have all the functionality like the regular registered users.
  - y
- Have additional privilege likes user account removals or passwordreset.
  - y

(___/1pt) (y/n) Do you have a bruteforce method as the comparison basis for the puzzle solver.

- y

(___/1pt) (y/n) Do you have a better algorithm than bruteforce.

- y

(___/1pt) Explain if the puzzle is targeted at a single user or multiplayer, competitive or noncompetitive.

- It is a single user game, and is noncompetitive. It can be competitive, like who can do it in less number of tries.

if it is a multiplayergame , address the possibility of the direct peertopeer communications without going through the host.

(___/1pt) Explain how to deploy your product.

1. Prerequisites:
Have Git installed.
Have a Heroku account and install the Heroku CLI.

2. Prepare Flask Application:
Create a Procfile with the command to run the app.
Ensure that there is a requirements.txt file listing all necessary packages.

3. Create a Heroku Application:
Log in to Heroku CLI.
Create a new Heroku app using `heroku create appname`.

4. Add a Database (if needed):
Add Heroku Postgres addon: `heroku addons:create herokupostgresql:hobbydev`.
Update the Flask app's configuration to use the provided DATABASE_URL environment variable.

5. Deploy Your Application:
Push the code to Heroku using `git push heroku main`

6. Ensure One Dyno is Running:
Scale the web process to one dyno: `heroku ps:scale web=1`.

7. Open the Application:
View the deployed application in a web browser using `heroku open`.