Lucky Unicorn Game

# Scenario

You have decided to create a fun game to raise money for the charity Doctors without Borders. You will set up your computer at lunch time and players will pay to play. Here are the rules…

Users pay an initial amount at the start of the game. The cost should be $1 per round and users should press <enter> to play. The computer should then generate a token that is either a zebra, horse, donkey, or unicorn. This should be displayed to the user. If the token is a unicorn, the user wins $5, if it is a zebra or horse, they win 50c and if it is a donkey then they do not win anything.

The maximum amount of money that students can spend on the game is $10 per session. The game should allow players to continue or quit provided they have not lost all their money. It should supply appropriate feedback so that the user knows how much money they have won or lost each round and how much money they have left.

Once students have no more money, the game should end (although players do have the option of quitting while they are ahead).

|  |
| --- |
| Variations (optional) Once you have a working game, you are welcome to develop the outcome further. Here are some variations you could consider…   * Change the tokens and context (so rather than having zebra, horses, donkeys, and unicorns the game could involve other items * Allow users to bet more than $1 per round and adjust the pay-outs proportionally <be careful to set up you game so that the house has a long-term advantage) |

## Task

1. Set up a project folder called “Lucky\_Unicorn” - for saving all files related to this this task
2. Decompose the problem (write down the decomposition on the template supplied)
3. For each part of the problem, write (and test) each piece of code. Before you write a piece of code, you should generate a quick test plan so that you can confirm that the code works correctly. Place your test plan and testing evidence on the supplied template.
4. Combine your code into a fully working program
5. Test and debug your program to ensure that it works for expected, boundary and unexpected values
6. Ask a friend / parent to play your game. Watch them as they do this and make note of any changes that could be made to make the game easier to use
7. Make the changes identified in the previous step
8. Retest your game to ensure that it still works correctly