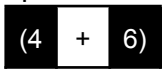
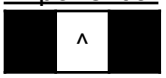
PUZZLE PITCH: BEDMAS

If you saw 6 numbers, had a selection of 5 operators to use, and knew the outcome, would you know what the equation to get the outcome looked like? BEDMAS is a new puzzle type designed to get users to answer this question for each of its puzzles. Along with being fun to solve, you'll be pleased to know that it also helps exercise users' understanding of fundamental arithmetic in an enjoyable and healthy way. An example of how the puzzle works is as shown in figure 1, where the black squares are the whole numbers to be operated on and the white squares are positions to assign each operator. Each operator position can only hold a single operator unless it is the brackets operator, in which case, one additional operator will be assigned to the same position. The six operators that may be used in this puzzle are as follows:

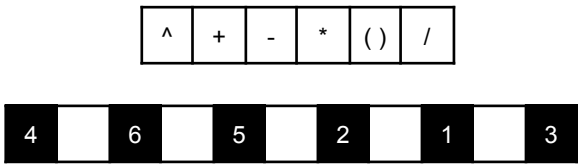
<table>
<tr>
<td>Addition
 
This operator adds the left number to the right number.</td>
<td>Subtraction
 
This operator subtracts the right number off the left number.</td>
</tr>
<tr>
<td>Multiplication
 
This operator multiples the left number by the right number.</td>
<td>Division
 
This operator divides the left number by the right number.</td>
</tr>
<tr>
<td>Brackets
 
This operator groups two numbers together whose additional common operator (which will be placed in the white square shown above) must be executed before any other operator in the array. For Example:
 
Is equivalent to:
 
The Brackets operator can only be placed on an operator square and not on a number square; that way it will enclose both left and right number positions.</td>
<td>Exponential
 
This operator raises the left number (left black square) to the power of the right number (right black square).</td>
</tr>
</table>

The initial array will always hold six numbers and 5 operator squares. The operators will execute in BEDMAS order, this means Brackets (B) will execute first, Exponential (E) second Division (D) third and so on. A selection of operators will be displayed and the user must use all these operators in the equation they produce. Once all operators have been moved into the desired position by the player, the player may submit. The equation that the player has created will execute and this will produce a number. If this number does not match the

desired output number, the user's score will increase by 1 and they may try again. The goal is for the player to score as low as possible. The desired number is always displayed below the number array and how it is produced is covered in the generating new puzzles section. When the player produces a number that does not match the desired output it will be displayed between the number array and the desired output number as shown in figure 2.

Examples of Initial and Completed Puzzle

Initial Puzzle
(figure 1)



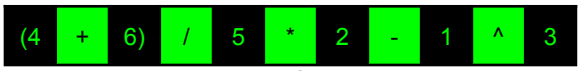
= 3

Example Wrong Answer
(figure 2)

= 46

= 3

Correct Answer
(figure 3)



= 3

(example of how you would fill in within a newspaper)

Generating new puzzles.

To generate new puzzles the number squares (black squares) of the initial array will be initialised to a random number between and including 1 and 9. A random selection of operators will then be randomly assigned to an operator position in BEDMAS order, however, there are a few conditions.

- Negative numbers are allowed but we don't want fractions or desired numbers whose absolute value is too high. Therefore when D is being randomly assigned, the right number must be a factor of the left number.
- When E is being assigned the right number cannot be negative and both the left and right number must be small enough such that the produced 'desired output number' is not too large.

Once the operators are assigned the resulting equation will be executed and the produced number 'stored' as the 'desired output number'. The operators will then be removed from the array and stored in a list. The array, the list of operators, and the desired output number will be displayed to the player to solve (figure 1). To make varying difficulties for this puzzle a few things will be done. These include adding more number squares in the array, hence, more occurrences of operators, including higher numbers in the array, allowing higher valued numbers to be involved in the exponential operator or changing what operators may be randomly selected for initial construction. For Example: At easier difficulties only the plus,

minus, multiply and divide operators can be used to construct a problem. Therefore these are the only operators the player would have to select from and think about. As the difficulty increases operators such as the exponential and brackets will be introduced. Excess operators could also be supplied in the operator list such that not all operators need to be used, therefore the player must carefully select which ones they will use. When generating a harder puzzle we would make sure that there is only a single solution unlike other difficulties where there may be multiple.

Why BEDMAS would be appealing in the app.

The core reason why we think users would enjoy BEDMAS is that the puzzle exercises the understanding of all aspects of basic arithmetic. By attempting these puzzle types, users will get to learn and understand how the order of operations (bedmas) affects an equation's outcome in a fun and enjoyable way. Away from why the BEDMAS puzzle type is appealing by itself, the reasons that we see BEDMAS being a great addition in an app form are as follows:

- For Users
    - We would utilise the touch screen by allowing the operators to all be drag and droppable
    - There would be a check answer button that works out the equation and gives the result in real time so that you can see the outcome of your attempt (if incorrect it would still print the answer that came from your equation).
    - There can be a scoring system for the amount of failed attempts as well as a timer to add an element of competitiveness to the puzzles.
- For Developers
    - Monetizing the app
        - Allowing excess operators means a system of swapping out operators could be added if the user sees a method of solving the puzzle that can't be done with the supplied operators.
    - Coding the generation of new puzzles would be straight forward
        - Generate numbers first then generate operators based on set rules to make the equation
        - The puzzle would be the numbers in order followed by the result of the original equation.
    - Coding for the check answer would be easy
        - You would just need to run the equation and check if it equals the original result.

Where The Puzzle Idea Originated and Licensing.

After thoroughly researching online to see if this puzzle exists, the conclusion was that this puzzle type is one of a kind. While there are similar types, it was an original idea from Joseph Sharratt that was helped to be further improved by Sam Gentry. Because of this, we believe that all rights to use the puzzle belong to them both (us) and so licensing the puzzle should be a relatively straightforward and simple process.