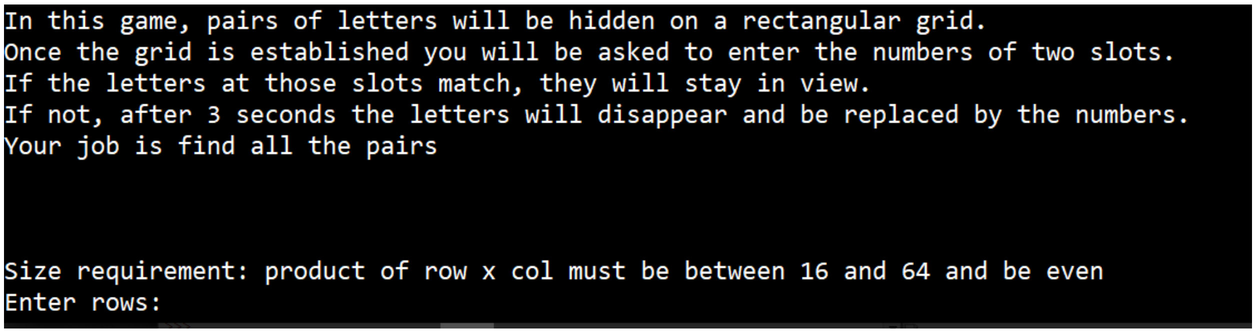
This assignment involves a vector of vectors, in essence a 2-dimensional vector.

You will be implementing a game of Match, where pairs of symbols will be hidden in a rectangular grid. The object of the game is to find the pairs of matching symbols. The symbols will come from the ASCII character set starting with A, B, C …

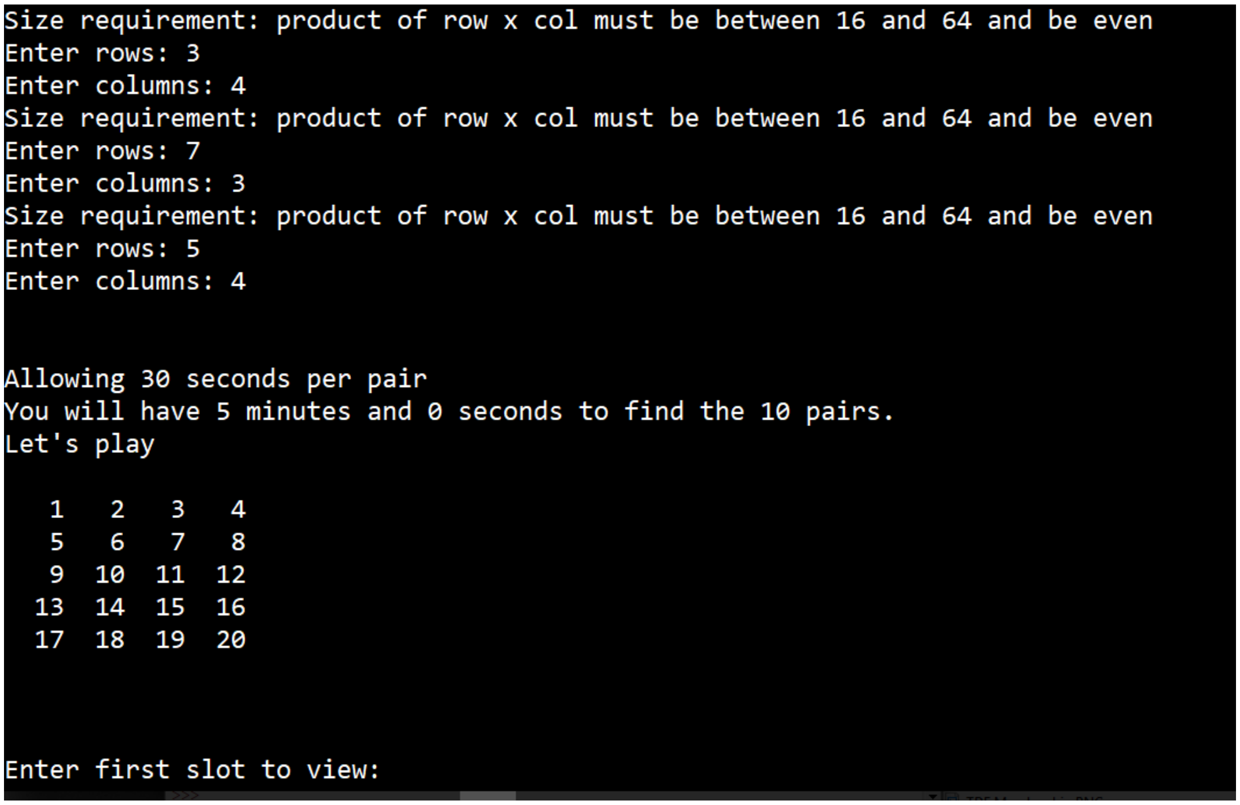
When the game starts, the following directions are displayed:



```
In this game, pairs of letters will be hidden on a rectangular grid.
Once the grid is established you will be asked to enter the numbers of two slots.
If the letters at those slots match, they will stay in view.
If not, after 3 seconds the letters will disappear and be replaced by the numbers.
Your job is find all the pairs



Size requirement: product of row x col must be between 16 and 64 and be even
Enter rows:
```

After the directions are displayed, the user is prompted to enter rows and columns for the rectangular grid. The value of the product of the rows and columns must be between 16 and 64. And since there will be pairs of symbols hidden, the product must be even number.

Entries for rows and columns that do not meet those criteria will cause a re-prompt of the "`Size requirement: …`" message. Such re-prompting is shown is the next display:

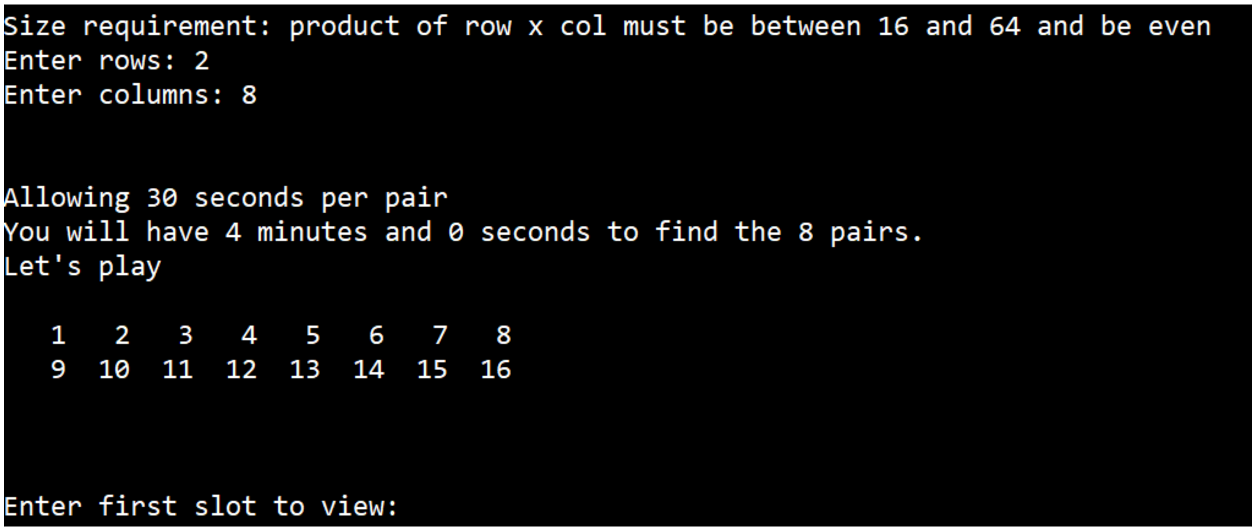


```
Size requirement: product of row x col must be between 16 and 64 and be even
Enter rows: 3
Enter columns: 4
Size requirement: product of row x col must be between 16 and 64 and be even
Enter rows: 7
Enter columns: 3
Size requirement: product of row x col must be between 16 and 64 and be even
Enter rows: 5
Enter columns: 4


Allowing 30 seconds per pair
You will have 5 minutes and 0 seconds to find the 10 pairs.
Let's play

   1   2   3   4
   5   6   7   8
   9  10  11  12
  13  14  15  16
  17  18  19  20



Enter first slot to view:
```

A grid with rows of 3 and columns of 4 is rejected because the product is too small. And a 7 x 3 grid is rejected because the product is odd. No special message is produced. Just issue the same prompts as before. A 5 row by 4 column grid meets the requirement. Since the product is twenty, the numbers 1 through 20 are displayed in a 5 row x 4 column grid. Hidden beneath those twenty slots will be ten pairs of symbols randomly deposited.

The product of rows x columns could be as large as 64, requiring 32 pairs of symbols. The first 26 pairs will use A through Z. The additional six symbols should come from those that follow Z on the ASCII chart:
`[\ ] ^ _ ``

If the number of rows was 2 and the number of columns was 8, the following grid would be displayed:



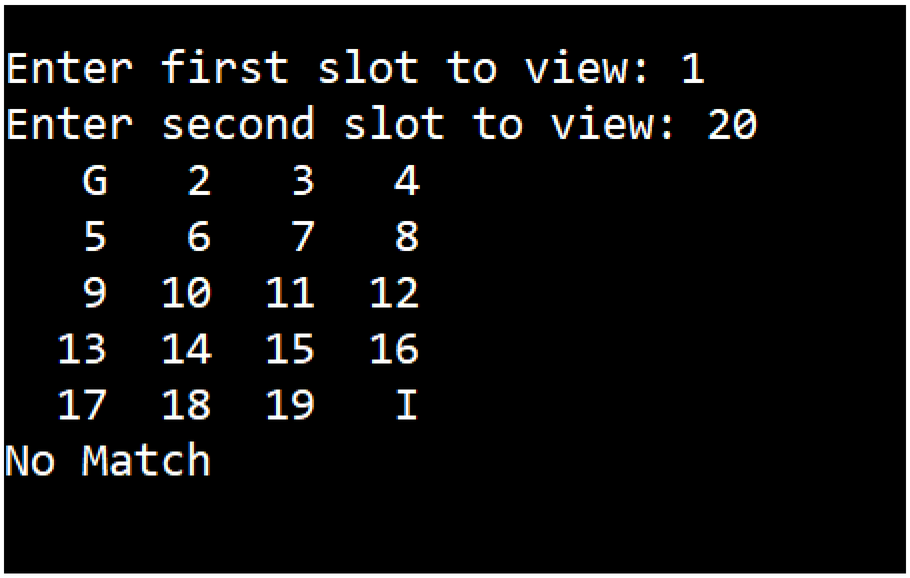
```
Size requirement: product of row x col must be between 16 and 64 and be even
Enter rows: 2
Enter columns: 8


Allowing 30 seconds per pair
You will have 4 minutes and 0 seconds to find the 8 pairs.
Let's play

   1   2   3   4   5   6   7   8
   9  10  11  12  13  14  15  16



Enter first slot to view:
```

All pictures from now on, will display a 5 row x 4 column grid.

The user is informed that he/she has 30 seconds per pair to find all the matches. With the5x4 grid, the user will have 300 seconds to match the ten pairs. It is indicated in an earlier display as
`You will have 5 minutes and 0 seconds to find the 10 pairs.`

The game ends if the user finds all the pairs within the allotted time or if time runs out.

Sometimes the last text of one display will be repeated at the top of the next display.

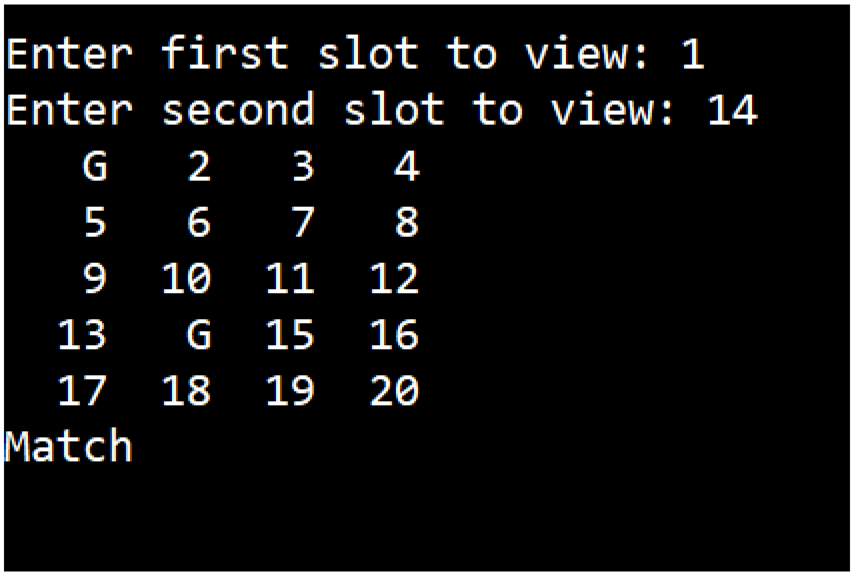


```
Enter first slot to view: 1
Enter second slot to view: 20
   G   2   3   4
   5   6   7   8
   9  10  11  12
  13  14  15  16
  17  18  19   I
No Match
```

The game asks the user to entire two slot numbers.

It is your job as a programmer to map slot numbers to row and column in the 2-D vector.

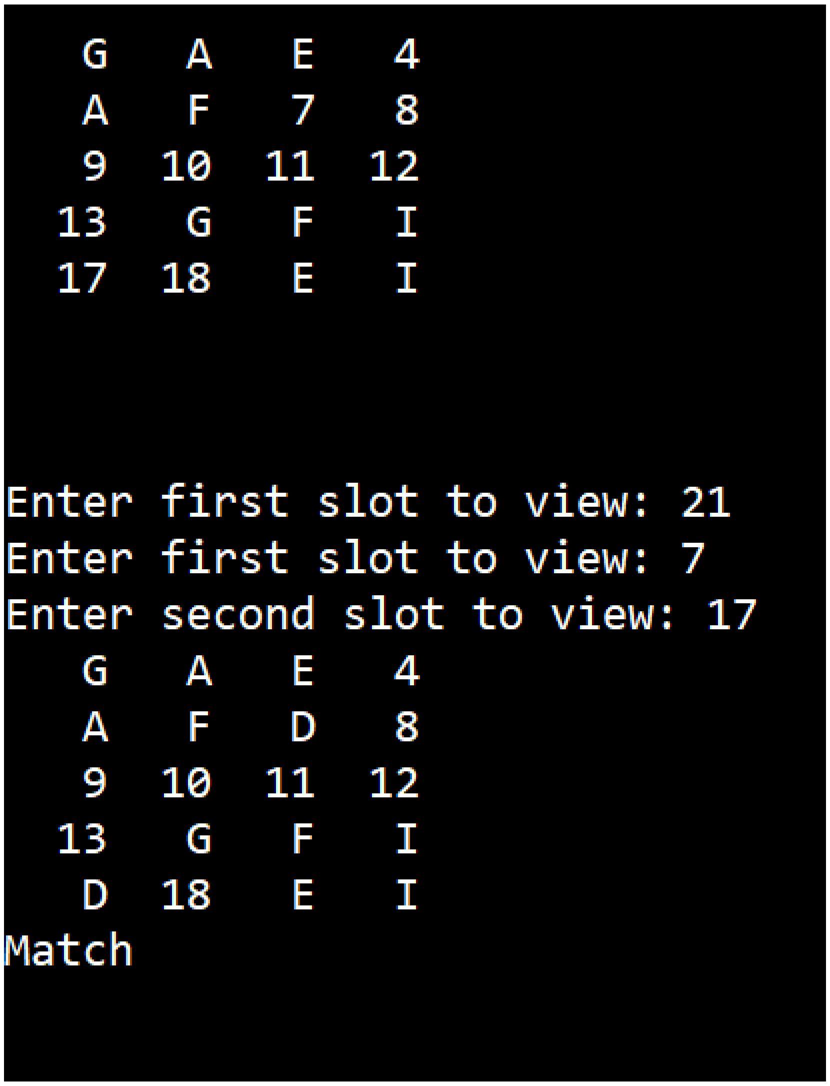
The program will display the symbols behind the numbered slots. If the symbols stored at those two slots are not identical, `No Match` is printed (as shown on the left). The revealed symbols stay in view for only 3 seconds, after which the screen is cleared by outputting 100 blank lines.



```
Enter first slot to view: 1
Enter second slot to view: 14
   G   2   3   4
   5   6   7   8
   9  10  11  12
  13   G  15  16
  17  18  19  20
Match
```

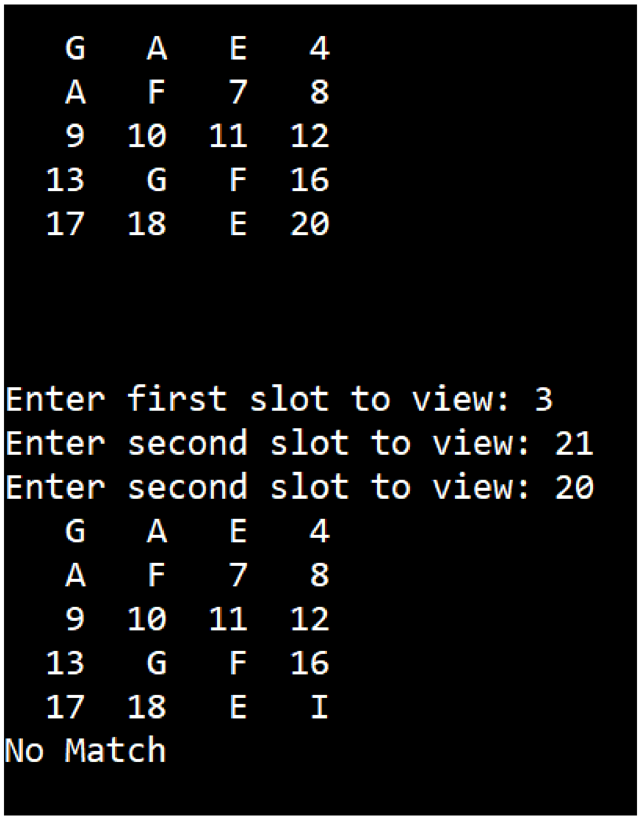
If the same symbol is stored at the two slots, the user will hear a bell (produced by '\a') and see `Match` printed beneath the grid. Then after 3 seconds the screen will be cleared, but when the screen is next shown the matched symbols will stay on the board having replaced the slot numbers.

Re-prompts will occur with the following situations:



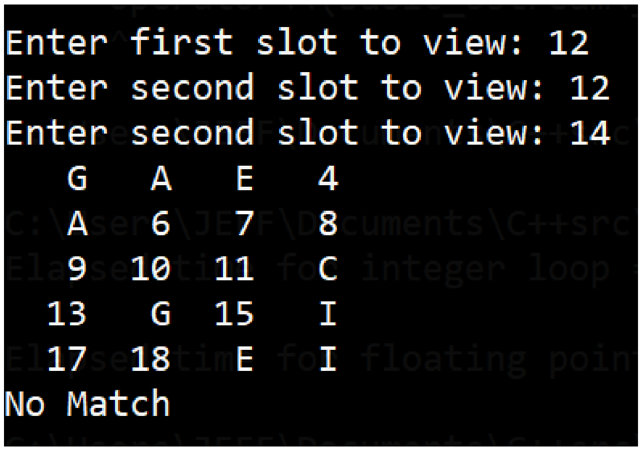
```
   G   A   E   4
   A   F   7   8
   9  10  11  12
  13   G   F   I
  17  18   E   I




Enter first slot to view: 21
Enter first slot to view: 7
Enter second slot to view: 17
   G   A   E   4
   A   F   D   8
   9  10  11  12
  13   G   F   I
   D  18   E   I
Match
```

When an out-of-bounds slot number is entered for either first or second slot number. For example a 5x4 grid has valid slot numbers between 1 to 20. In the figure to the left, the value of 21 is invalid as the first slot number.



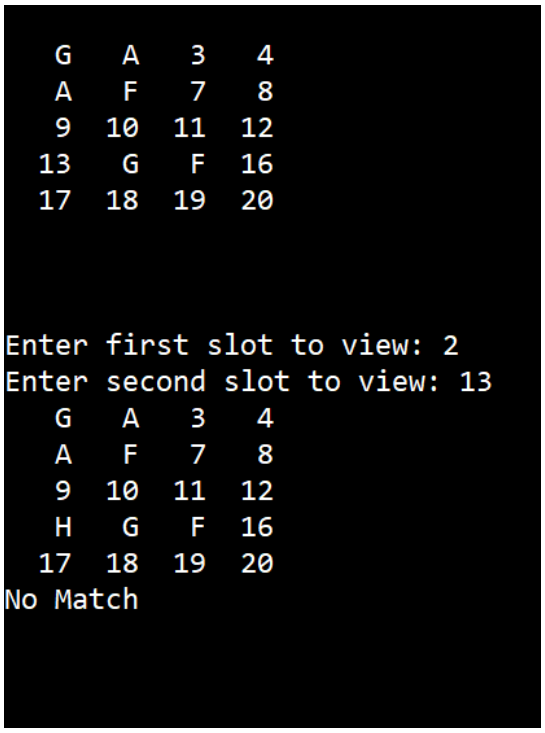
```
   G   A   E   4
   A   F   7   8
   9  10  11  12
  13   G   F  16
  17  18   E  20



Enter first slot to view: 3
Enter second slot to view: 21
Enter second slot to view: 20
   G   A   E   4
   A   F   7   8
   9  10  11  12
  13   G   F  16
  17  18   E   I
No Match
```

Here, there is a re-prompt for the second slot number.



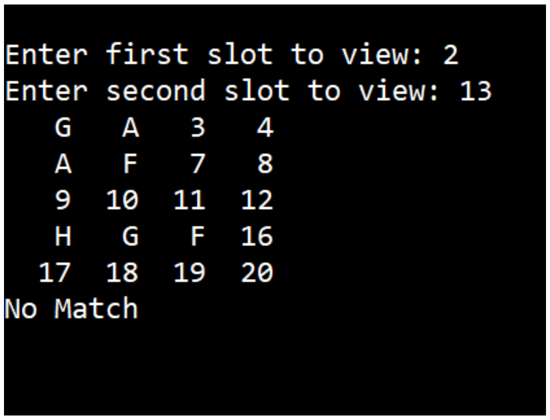
```
Enter first slot to view: 12
Enter second slot to view: 12
Enter second slot to view: 14
   G   A   E   4
   A   6   7   8
   9  10  11   C
  13   G  15   I
  17  18   E   I
No Match
```

A re-prompt should also be issued if the second slot number is identical to the first slot number.



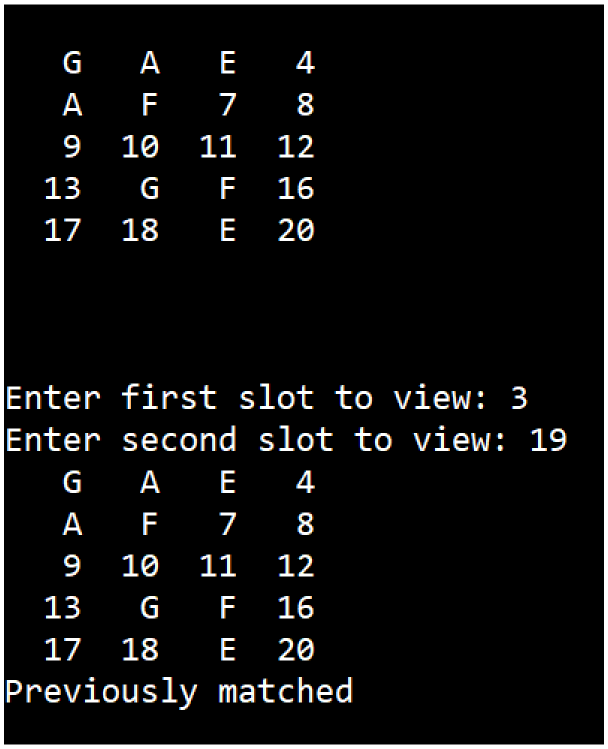
```
   G   A   3   4
   A   F   7   8
   9  10  11  12
  13   G   F  16
  17  18  19  20



Enter first slot to view: 2
Enter second slot to view: 13
   G   A   3   4
   A   F   7   8
   9  10  11  12
   H   G   F  16
  17  18  19  20
No Match
```

There is **no** re-prompting if a slot numbers has already been revealed. In this case, slot 2 was earlier unmasked.



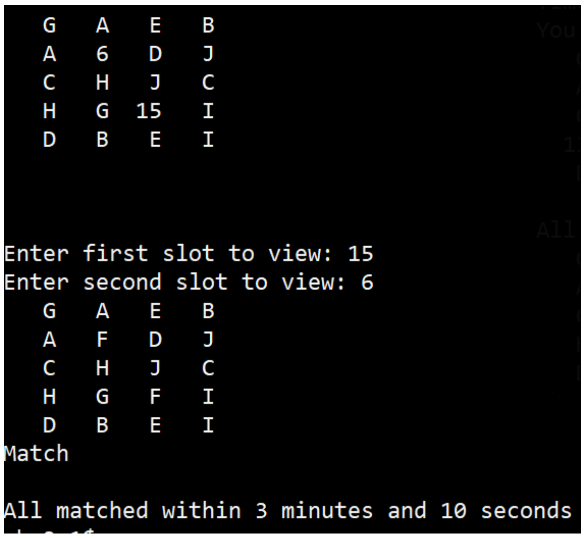
```
Enter first slot to view: 2
Enter second slot to view: 13
   G   A   3   4
   A   F   7   8
   9  10  11  12
   H   G   F  16
  17  18  19  20
No Match
```

The same thing applies if both slots have been already revealed. Here both slot 2 and 13 have been revealed earlier. No re-prompting should be done, and if these two cells are not the same, `No Match` should be displayed.



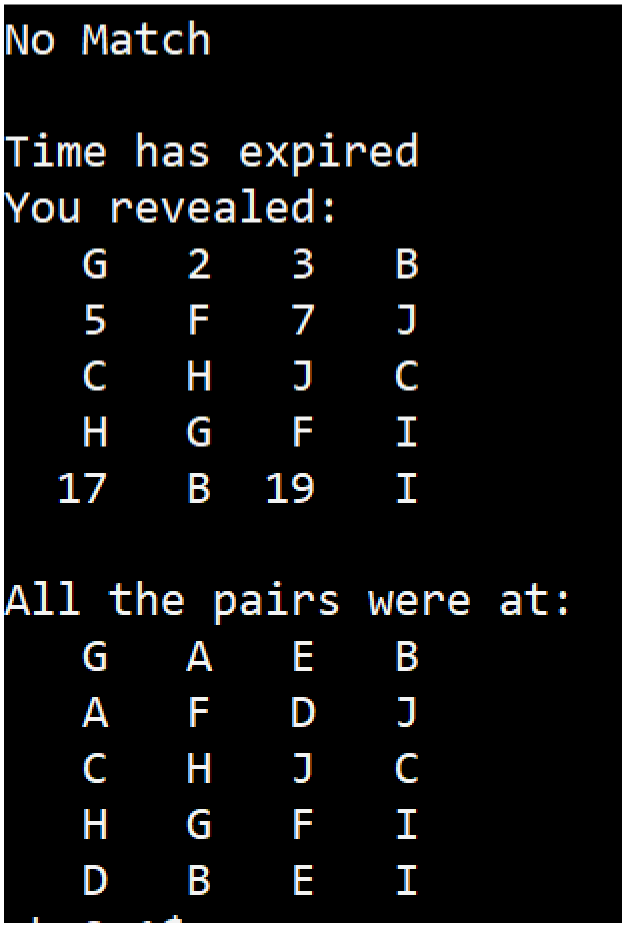
```
   G   A   E   4
   A   F   7   8
   9  10  11  12
  13   G   F  16
  17  18   E  20



Enter first slot to view: 3
Enter second slot to view: 19
   G   A   E   4
   A   F   7   8
   9  10  11  12
  13   G   F  16
  17  18   E  20
Previously matched
```

But if the two slot numbers have been both earlier revealed and are identical, the message `Previosly matched` shoud be printed.



```
   G   A   E   B
   A   6   D   J
   C   H   J   C
   H   G  15   I
   D   B   E   I



Enter first slot to view: 15
Enter second slot to view: 6
   G   A   E   B
   A   F   D   J
   C   H   J   C
   H   G   F   I
   D   B   E   I
Match

All matched within 3 minutes and 10 seconds
```

The game proceeds until either the user finds all the pairs or time runs out. Shown here is when the user beats the clock. And the last thing that is printed before the game ends is the message shown that indicated the amount of time in minutes and seconds that it took to find all the matches.



```
No Match

Time has expired
You revealed:
   G   2   3   B
   5   F   7   J
   C   H   J   C
   H   G   F   I
  17   B  19   I

All the pairs were at:
   G   A   E   B
   A   F   D   J
   C   H   J   C
   H   G   F   I
   D   B   E   I
```

If the allotted time expires, the positions of all the symbols are revealed and then the game terminates.