1. MetaverseX Copyright. We will find the arrangement of the mobius strip such that all letters pertianing to pentominoes fit on a mobius strip of size 15x4.

⟨ Include libraries. 2 ⟩;
⟨ Set definitions. 3 ⟩
⟨ Set global variables. 4 ⟩
⟨ Generate sequences in the table for the exact colored coloring problem. 6 ⟩
⟨ Backtracking on generated table. 33 ⟩

2. First we include libraries.

⟨ Include libraries. 2 ⟩ ≡
```
#include <stdio.h>
```
This code is used in section 1.

3. `MAX_ARRAY` is the maximum number of placings a letter pentomino can have on the 154 mobius strip.

⟨ Set definitions. 3 ⟩ ≡
```
#define MAX_ARRAY 1000
```
This code is used in section 1.

4. And set the global variables.

⟨ Set global variables. 4 ⟩ ≡
```
int l = 15, w = 4;
```
This code is used in section 1.

5. Then we generate all the places where each letter can be. O 11 12 13 14 15 O 12 13 14 15 16 O 13 14 15 16 17 O 14 15 16 17 18 ... O 1f 11 12 13 14 O 21 22 23 24 25 ... Z ... The formula for O is $\mathrm{ab}_1, ab_2, ab_3, ab_4, ab_5 for a in 1-4, b_1 = c for c from 1-15 and b_(i+1) = b_i + 1 (mod 15) for 1 < i <= 15$.

6. We generate the coordinates in the array that can be thought of as sequences corresponding to the letter as the first element in the sequence for the exact covering problem followed by the coordinates of the five perpendicularly adjacent pentominoes.

⟨ Generate sequences in the table for the exact colored coloring problem. 6 ⟩ ≡
  ⟨ O 7 ⟩;
  ⟨ P 15 ⟩;
  ⟨ Q 20 ⟩;

This code is used in section 1.

7. O is for setting the 3d array O with the five coordinates. These correspond to all placements of the pentomino on the mobius strip.

```
⟨ O 7 ⟩ ≡
  int O[MAX_ARRAY][5][2], c = 0;
  for (⟨ All the width 10 ⟩;
  ) {
    for (⟨ All x position in width 11 ⟩;
    ) {
      ⟨ Set O 8 ⟩
      ⟨ Increment c 12 ⟩;
    }
  }
  ⟨ Set sentinel for O. 13 ⟩
```

This code is used in section 6.

8. Setting all the coordinates and the positions of O.

```
⟨ Set O 8 ⟩ ≡
  for (⟨ All five consecutive boxes for O 9 ⟩
  ) {
    O[c][b][0] = (x + b) % 15;
    ⟨ If x+b is less than or equal to 15. 14 ⟩
    O[c][b][1] = y;
    else O[c][b][1] = w − y + 1
  }
```

This code is used in section 7.

9. Iteration for the consecutive horizontal boxes for O.

```
⟨ All five consecutive boxes for O 9 ⟩ ≡
  int b = 0;
  b < 5; b++
```

This code is used in section 8.

10. Iterating across the width of the mobius strip. Only relevant if the width of the pentomino is two.

```
⟨ All the width 10 ⟩ ≡
  int y = 0;
  y < 4; y++
```

This code is used in section 7.

11. Iterating across the length 15 of the mobius strip.

```
⟨ All x position in width 11 ⟩ ≡
  int x = 0;
  x < l; x++
```

This code is used in sections 7, 15, 20, and 26.

12. Increment c to put the next sequence of pentomino coordinates in the 3d array.

```
⟨ Increment c 12 ⟩ ≡
  c++;
```

This code is used in sections 7, 15, 20, and 26.

13. We've to set a sentinel in the array O to mark the end of positions of O in the mobius strip.

⟨ Set sentinel for O. 13 ⟩ ≡
```
O[c][0][0] = −1;
```
This code is used in section 7.

14. To deal with the overflow for the mobius strip to mirror back the Q across x axis.

⟨ If x+b is less than or equal to 15. 14 ⟩ ≡
```
if (x + b < 15)
```
This code is used in sections 8, 23, 25, 27, and 29.

15. P is four boxes and a tail. We need all rotations of P.

```
⟨ P 15 ⟩ ≡
  c = 0;
  for (⟨ 0 and 1 y coordinates. 18 ⟩;
  )
    for (⟨ All x position in width 11 ⟩;
    ) {
      ⟨ Set P right 16 ⟩
      ⟨ Increment c 12 ⟩;
      ⟨ Set P left 17 ⟩
      ⟨ Increment c 12 ⟩;
    }
  ⟨ Set sentinel for P. 19 ⟩;
  ⟨ If x+1 is less than 15 32 ⟩= if (x + 1 < 15)
```

This code is used in section 6.

16. We set P in the array P where P's tail is on the right-hand side that is it is the mirror image across y axis of what looks like a P pentameter configuration

```
⟨ Set P right 16 ⟩ ≡
  P[c][0][0] = x;
  P[c][0][1] = y;
  P[c][1][0] = (x + 1) % 15;
  ⟨ If x+1 is less than 15 32 ⟩
  P[c][1][1] = y;
  else P[c][1][1] = w − y + 1;
  P[c][2][0] = x;
  P[c][2][1] = y + 1;
  P[c][3][0] = (x + 1) % 15;
  ⟨ If x+1 is less than 15 32 ⟩
  P[c][3][1] = y + 1;
  else P[c][3][1] = w − (y + 1) + 1;
  P[c][4][0] = (x + 1) % 15;
  ⟨ If x+1 is less than 15 32 ⟩
  P[c][4][1] = y + 2;
  else P[c][4][1] = w − (y + 2) + 1;
```

This code is used in section 15.

17. We set P in the array P where P's tail is on the left-hand side that is it is the mirror image across y axis of what looks like a P pentameter configuration.

```
⟨ Set P left 17 ⟩ ≡
  P[c][0][0] = x;
  P[c][0][1] = y;
  P[c][1][0] = (x + 1) % 15;
  ⟨ If x+1 is less than 15 32 ⟩
  P[c][1][1] = y;
  else P[c][1][1] = w − y + 1;
  P[c][2][0] = x;
  P[c][2][1] = y + 1;
  P[c][3][0] = (x + 1) % 15;
  ⟨ If x+1 is less than 15 32 ⟩
  P[c][3][1] = y + 1;
  else P[c][3][1] = w − (y + 1) + 1;
  P[c][4][0] = x;
  P[c][4][1] = y + 2;
```

This code is used in section 15.

18. For the pentaminoes that have width of 3

```
⟨ 0 and 1 y coordinates. 18 ⟩ ≡
  int y = 0;
  y < 2; y++
```

This code is used in sections 15 and 20.

19. We set a sentinel to mark the end of P positions in the array P.

```
⟨ Set sentinel for P. 19 ⟩ ≡
  P[c][0][0] = −1;
```

This code is used in section 15.

20. Arranging pentomines for Q. We one by one proceed to put all rotations of Q across all positions.

```
⟨ Q 20 ⟩ ≡
  c = 0;
  int Q[MAX_ARRAY][5][2];
  for (⟨ 0,1,2 y coordinates 21 ⟩;
  ) {
    for (⟨ All x position in width 11 ⟩;
    ) {
      ⟨ Set Q bottom right 23 ⟩
      ⟨ Increment c 12 ⟩;
      ⟨ Set Q bottom left 25 ⟩
      ⟨ Increment c 12 ⟩;
    }
  }
  for (⟨ 0 and 1 y coordinates. 18 ⟩;
  ) {
    for (⟨ All x position in width 11 ⟩;
    ) {
      ⟨ Set Q top left 29 ⟩
      ⟨ Increment c 12 ⟩;
      ⟨ Set Q bottom left 25 ⟩
      ⟨ Increment c 12 ⟩;
    }
  }
  ⟨ Set Q top left 29 ⟩= for ⟨ Four consecutive vertical boxes for Q 30 ⟩;
  ⟨ If x minus one is less than one 31 ⟩Q[c][0][0] = (x − 1) % 15;
  Q[c][0][1] = y − 1;
  ⟨ Set sentinel for Q. 26 ⟩;
```

This code is used in section 6.

21. For pentominoes that have width of 2.

```
⟨ 0,1,2 y coordinates 21 ⟩ ≡
  int y = 0;
  y < 3; y++
```

This code is used in section 20.

22. For pentominoes that have width 2 and major horizontal center is one below the top extremity.

```
⟨ 2,3,4 y coordinates 22 ⟩ ≡
  int y = 1;
  y < 4; y++
```

This code is used in section 26.

23. Similarly, we have 4 consecutive horizontal boxes and one box is below the third box.

```
⟨ Set Q bottom right 23 ⟩ ≡
  for (⟨ Four consecutive boxes for Q 24 ⟩
  ) {
    Q[c][b][0] = (x + b) % 15;
    ⟨ If x+b is less than or equal to 15. 14 ⟩
    Q[c][b][1] = y;
    else Q[c][b][1] = w − y + 1;
  }
  Q[c][4][0] = (x + 2) % 15;
  ⟨ If x+1 is less than 15 32 ⟩
  Q[c][4][1] = y + 1;
  else Q[c][4][1] = w − (y + 1) + 1;
```

This code is used in section 20.

24. The pentomino that is 4 consecutive horizontal boxes and one box is below the second box.

```
⟨ Four consecutive boxes for Q 24 ⟩ ≡
  int b = 0;
  b < 4; b++
```

This code is used in sections 23, 25, 27, and 29.

25. Set a Q where Q's main line is horizontally aligned and the appendage is on the bottom left.

```
⟨ Set Q bottom left 25 ⟩ ≡
  for (⟨ Four consecutive boxes for Q 24 ⟩;
  ) {
    Q[c][b][0] = (x + b) % 15;
    ⟨ If x+b is less than or equal to 15. 14 ⟩
    Q[c][b][1] = y;
    else Q[c][b][1] = w − y + 1;
  }
  Q[c][4][0] = (x + 1) % 15;
  ⟨ If x+1 is less than 15 32 ⟩
  Q[c][4][1] = y + 1;
  else Q[c][4][1] = w − (y + 1) + 1;
```

This code is used in section 20.

26. Setting the sentinel so that we know when the number of placements terminate in the array Q.

```
⟨ Set sentinel for Q. 26 ⟩ ≡
  Q[c][0][0] = −1;
  for (⟨ 2,3,4 y coordinates 22 ⟩;
  ) {
    for (⟨ All x position in width 11 ⟩;
    ) {
      ⟨ Set Q top right 27 ⟩
      ⟨ Increment c 12 ⟩;
      ⟨ Set Q top left 29 ⟩
      ⟨ Increment c 12 ⟩;
    }
  }
```

This code is used in section 20.

27. We flip the Q bottom left across x axis.

```
⟨ Set Q top right 27 ⟩ ≡
  for (⟨ Four consecutive boxes for Q 24 ⟩;
  ) {
    Q[c][b][0] = (x + b) % 15;
    ⟨ If x+b is less than or equal to 15. 14 ⟩;
    Q[c][b][1] = y;
    else Q[c][b][1] = w − y + 1;
  }
  Q[c][4][0] = (x + 2) % 15;
  ⟨ If x+2 is less than or equal to 15 28 ⟩;
  Q[c][4][1] = y + 1;
  else Q[c][4][1] = w − (y + 1) + 1;
```

This code is used in section 26.

28. So that the overflow in the pentomino jigsaw satisfies the mobius strip property.

⟨ If x+2 is less than or equal to 15 28 ⟩ ≡
```
if (x + 2 ≤ 15)
```
This code is used in section 27.

29. We flip Q bottom left across x axis.

⟨ Set Q top left 29 ⟩ ≡
```
for (⟨ Four consecutive boxes for Q 24 ⟩;
) {
  Q[c][b][0] = (x + b) % 15;
  ⟨ If x+b is less than or equal to 15. 14 ⟩;
  Q[c][b][1] = y;
  else Q[c][b][1] = w − y + 1;
}
Q[c][4][0] = (x + 1) % 15;
⟨ If x+1 is less than 15 32 ⟩;
Q[c][4][1] = y − 1;
else Q[c][4][1] = w − (y − 1) + 1;
```
This code is used in sections 20 and 26.

30. We need four consecutive vertical boxes for Q along Q's main line.

⟨ Four consecutive vertical boxes for Q 30 ⟩ ≡
```
int y = 0;
y < 4; y++
```
This code is used in section 20.

31. We need that x -1 is less than one

⟨ If x minus one is less than one 31 ⟩ ≡
```
if (x − 1 < 1)
```
This code is used in section 20.

32. We need that x -1 is less than 15

⟨ If x+1 is less than 15 32 ⟩ ≡
```
if (x + 1 < 15)
```
This code is used in sections 15, 16, 17, 23, 25, and 29.

33. Then we apply backtracking in which each pentomino letter represents a depth l in the tree and P$_l$ *stands for the set of x coordinates in extended hex notation and y coordinates in the strip that the letter occupies.* ∎

⟨ **Backtracking on generated table.** 33 ⟩ ≡

;

This code is used in section 1.

b: 9, 24.
c: 7.
l: 4.
MAX_ARRAY: 3, 7, 20.
O: 7.
Q: 20.
w: 4.
x: 11.
y: 10, 18, 21, 22, 30.

⟨ 0 and 1 y coordinates. 18 ⟩ Used in sections 15 and 20.
⟨ 0,1,2 y coordinates 21 ⟩ Used in section 20.
⟨ 2,3,4 y coordinates 22 ⟩ Used in section 26.
⟨ All five consecutive boxes for O 9 ⟩ Used in section 8.
⟨ All the width 10 ⟩ Used in section 7.
⟨ All x position in width 11 ⟩ Used in sections 7, 15, 20, and 26.
⟨ Backtracking on generated table. 33 ⟩ Used in section 1.
⟨ Four consecutive boxes for Q 24 ⟩ Used in sections 23, 25, 27, and 29.
⟨ Four consecutive vertical boxes for Q 30 ⟩ Used in section 20.
⟨ Generate sequences in the table for the exact colored coloring problem. 6 ⟩ Used in section 1.
⟨ If x minus one is less than one 31 ⟩ Used in section 20.
⟨ If x+1 is less than 15 32 ⟩ Used in sections 15, 16, 17, 23, 25, and 29.
⟨ If x+2 is less than or equal to 15 28 ⟩ Used in section 27.
⟨ If x+b is less than or equal to 15. 14 ⟩ Used in sections 8, 23, 25, 27, and 29.
⟨ Include libraries. 2 ⟩ Used in section 1.
⟨ Increment c 12 ⟩ Used in sections 7, 15, 20, and 26.
⟨ O 7 ⟩ Used in section 6.
⟨ P 15 ⟩ Used in section 6.
⟨ Q 20 ⟩ Used in section 6.
⟨ Set O 8 ⟩ Used in section 7.
⟨ Set P left 17 ⟩ Used in section 15.
⟨ Set P right 16 ⟩ Used in section 15.
⟨ Set Q bottom left 25 ⟩ Used in section 20.
⟨ Set Q bottom right 23 ⟩ Used in section 20.
⟨ Set Q top left 29 ⟩ Used in sections 20 and 26.
⟨ Set Q top right 27 ⟩ Used in section 26.
⟨ Set definitions. 3 ⟩ Used in section 1.
⟨ Set global variables. 4 ⟩ Used in section 1.
⟨ Set sentinel for O. 13 ⟩ Used in section 7.
⟨ Set sentinel for P. 19 ⟩ Used in section 15.
⟨ Set sentinel for Q. 26 ⟩ Used in section 20.

MOBIUS