3. Warcaby

Zadanie

W templatce programu jest zdefiniowana tablica dwuwymiarowa `board[20][20]`.

Warcaby

Uzupełnij funkcje, których argumentem jest dwuwymiarowa tablica `board[][]` reprezentująca planszę do gry w warcaby.

1. Funkcja `start(char[N][], int n)` ustawia bierki w ich początkowych położeniach. Bierki powinny być ustawione wyłącznie na ciemnych polach. Zakładamy, że pole o współrzędnych `board[0][0]` jest ciemne. Bierki gracza pierwszego są oznaczane na planszy znakiem '1' i zajmują rzędy począwszy od ineksu 0 (rosnąco), a bierki gracza drugiego oznaczamy znakiem '2' i początkowo zajmują rzędy od `n - 1` (malejąco). Bierki każdego z graczy zajmują kolejne 4 rzędy (czyli każdy gracz ma początkowo $2n$ bierek).

2. Funkcja `int move(char board[][N], int n, int i, int j))` sprawdza, czy bierka na pozycji `board[i][j]` może wykonać ruch (czyli czy pole docelowe jest puste) i jeżeli tak, to wykonuje ten ruch. Jeżeli możliwy jest ruch w obie strony (po skosie w lewo i po skosie w prawo) to wykonywany jest ruch w stronę indeksu `[j+1]`. Bierki mogą wykonywać ruchy wyłącznie po skosie do przodu (czyli bierka na polu `[i][j]` może przejść na pole `[i+1][j+1]` lub `[i+1][j-1]` (dla gracza '1') i na pole `[i-1][j+1]` lub `[i-1][j-1]` (dla gracza '2'). Funkcja zwraca wartość 1 jeżeli ruch został wykonany i 0 jeżeli wyznaczona bierka nie mogła wykonać ruchu.

3. Funkcja `int capture(char board[][N], int n, int i, int j))` sprawdza, czy bierka na pozycji `board[i][j]` może zbić bierkę przeciwnika (czyli czy jakaś bierka przeciwnika zajmuje sąsiednie pole po skosie do przodu i czy za tą bierką w tym samym kierunku jest pole puste). Jeżeli tak, to wykonuje bicie, czyli przeskakuje o 2 pozycje do przodu i dwie pozycje w prawo lub w lewo, a przeskoczona bierka przeciwnika jest usuwana z planszy. Jeżeli możliwe jest bicie w obie strony (po skosie w lewo i po skosie w prawo) to wykonywane jest bicie w stronę indeksu `[j+2]`. Podobnie jak w przypadku ruchu, bierki mogą zbijać bierki przeciwnika wyłącznie po skosie do przodu (czyli bierka na polu `[i][j]` może przejść na pole `[i+2][j+2]` lub `[i+2][j-2]` (dla gracza '1') i na pole `[i-2][j+2]` lub `[i-2][j-2]` (dla gracza '2'). Funkcja zwraca wartość 1 jeżeli bicie zostało wykonane i 0 jeżeli wyznaczona bierka nie mogła wykonać zbicia.

4. Funkcja `void print_board(char board[][N], int n)` wypisuje stan planszy zgodnie z rosnącą wartością indeksów. Puste pola mają zawierać znak podkreślnika '_'. Po każdym znaku pola powinna znajdować się spacja.

Przebieg gry (zrealizowany w funkcji `main()` templatki): Wykonywana jest zadana liczba kroków. Gracze wykonują kroki naprzemiennie. Krok polega na:

1. Losowaniu pola planszy, aż to momentu gdy wylosowane zostanie pole, na krórym znajduje się bierka gracza aktualnie wykonującego ruch.

2. Gracz sprawdza czy może wykonać bicie i jeżeli tak, to je wykonuje (funkcja `capture()`).

3. Jeżeli nie jest możliwe bicie to gracz próbuje wykonać normaly ruch (funkcja `move()`). Jeżeli ruch również nie jest możliwy, przechodzimy do następnego kroku.

Wejście
Trzy liczby całkowite: rozmiar planszy – parzysta liczba $10 \leqslant n \leqslant 16$, początkowa wartość ziarna generacji dla funkcji `srand()` i liczba kroków do wykonania.

Wyjście
Stan planszy po wykonaniu zadanej liczby kroków oraz liczba bierek, jaka pozostała każdemu z graczy.

Przykład:

Wejście:
```
10 2 100
```

Wyjście:

```
_ _ 1 _ 1 _ _ _ 1 _
_ _ _ 1 _ _ _ 1 _ 1
_ _ _ _ 1 _ 1 _ 1 _
_ 1 _ _ _ 1 _ 2 _ 1
1 _ _ _ 1 _ 1 _ _ _
_ 1 _ 2 _ 1 _ 2 _ _
2 _ 2 _ 2 _ 2 _ 2 _
_ _ _ _ _ 2 _ 2 _ 2
2 _ 2 _ 2 _ 2 _ 2 _
_ _ _ 2 _ _ _ _ _ 2
17 18
```