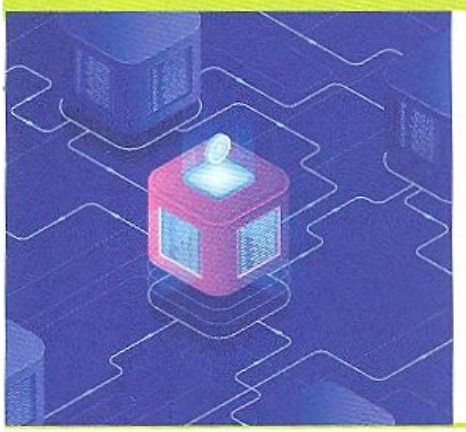


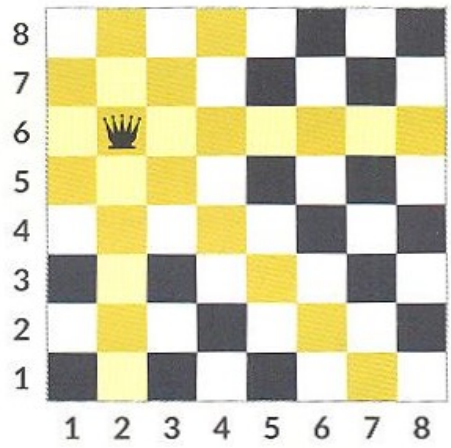
15. Definiowanie funkcji obliczeniowych

- Wykorzystanie instrukcji warunkowej w obliczeniach
- Instrukcja iteracji
- Analizowanie i testowanie rozwiązań

ZADANIE NA START

Ile pól atakuje hetman

Hetman (królowa) porusza się w tym samym rzędzie lub w tej samej kolumnie albo po przekątnych, w dowolnym kierunku, o dowolną liczbę niezajętych pól. Pola, do których figura może dotrzeć w jednym ruchu, są przez nią atakowane. Hetman stojący na szachownicy 8 × 8 na polu (2, 6), gdzie 2 oznacza numer kolumny, a 6 – numer wiersza, atakuje 14 + 9 = 23 pola. Określ liczbę atakowanych pól na szachownicy 32 × 32, gdy hetman znajduje się na polu (25, 30).



SZACHY, SZTUCZNA INTELIGENCJA I PROGRAMOWANIE

Najstarsza maszyna, która potrafiła w ograniczonym stopniu grać w szachy, powstała już pod koniec XIX w. Obecnie programiści tworzą coraz skuteczniejsze algorytmy, pozwalające maszynie zmieścić się w regulaminowym czasie rozgrywania partii szachowych, rozpatrujące pozycje, które wpłyną i nie wpłyną na ocenę sytuacji, wybierające kilka najbardziej obiecujących ruchów i znajdujące prawdopodobnie najlepszy ruch. W zależności od złożoności problemy szachowe mogą wymagać zastosowania prostej funkcji z parametrem i operatorów arytmetycznych albo bardziej zaawansowanych narzędzi.

POSZUKAJ W INTERNECIE

Poszukaj w internecie informacji na temat pojedynku Garriego Kasparowa ze stworzonym przez IBM systemem Deep Blue oraz innych pojedynków komputer vs. człowiek.

W tabeli przedstawiono podstawowe konstrukcje stosowane w języku Python.

Konstrukcja	Składnia	Przykład
trukcja przypisania	`nazwa_zmiennej = wartość`	`x = 3`
trukcja warunkowa sta	`if warunek:` `  instrukcje`	`if x % 3 == 0:` `  print("liczba podzielna przez 3")`

rukcja warunkowa żona	`if warunek:`
	`  instrukcje`
	`elif warunek:`
	`  instrukcje`
	`else:`
	`  instrukcje`
		`if x > 0:`
		`  print("liczba dodatnia")`
		`elif x == 0:`
		`  print("zero")`
		`else:`
		`  print("liczba ujemna")`

a for	`for nazwa_zmiennej in sekwencja:` `  instrukcje`	`for i in range(10):` `  print(i)`

a while	`while warunek:`
	`  instrukcje`
		`while x > 10:`
		`  print(x)`
		`  x = x - 1`

kcja	`def nazwa(parametry):` `  instrukcje`	`def pole_prostokata(bok1, bok2):` `  return bok1 * bok2`

Ćwiczenie 1. Ile pól ma szachownica

Szachownica złożona jest z równych pól, naprzemiennie białych i czarnych. Przeanalizuj dane w tabeli i zdefiniuj funkcję **ile_pol(n)**, której parametrem jest rozmiar szachownicy, a wynikiem liczba pól na szachownicy. Parametr **n** może przyjmować wartości od 2 do 32.

Wywołanie funkcji	Wynik
ile_pol(8)	**64**
ile_pol(13)	**169**

- Plansza złożona jest z $n \times n$ pól, a więc $8 \cdot 8 = 64$, $13 \cdot 13 = 169$.
- Zdefiniuj funkcję **ile_pol(n)**.

```
1. def ile_pol(n):
2.     return n * n
```

- Sprawdź działanie skryptu z różnymi danymi.

Pamiętaj, aby następne ćwiczenia wykonywać w tym samym pliku.

INSTRUKCJA WARUNKOWA

Instrukcja warunkowa pozwala na wykonanie różnych obliczeń w zależności od tego, czy zdefiniowane wyrażenie logiczne jest prawdziwe, czy fałszywe.

- Jeśli liczba jest parzysta, podziel wartość przez 2, w przeciwnym przypadku odejmij 1.
- Jeśli liczba jest dodatnia, przypisz zmiennej znak 1, gdy ujemna przypisz –1, w przeciwnym przypadku przypisz 0.

Instrukcja warunkowa może składać się z jednej, dwóch lub trzech części.

- Na początku zawsze znajduje się słowo kluczowe **if** oraz warunek lub wyrażenie, które mogą mieć wartość **prawda** (1 lub inna wartość) lub **fałsz** (0). Wiersz należy zakończyć dwukropkiem.
- Potem następuje instrukcja lub ciąg instrukcji do wykonania w przypadku spełnienia warunku.
- Słowo kluczowe **elif** (skrót od *else if*) wprowadza nowy warunek lub wyrażenie oraz instrukcje do wykonania w przypadku spełnienia tego warunku.
- Słowo kluczowe **else** wprowadza instrukcję do wykonania, gdy żadne z powyższych warunków nie są prawdziwe.

ZAPAMIĘTAJ!

```
if <wyrażenie jest prawdziwe>:
     <wykonaj instrukcję 1>
elif <poprzednie wyrażenie jest fałszywe, ale to jest prawdziwe>:
     <wykonaj instrukcję 2>
else:
     <wykonaj instrukcję 3>
```

Instrukcja warunkowa na rys. 1 składa się z dwóch części. Można ją odczytać następująco:

- przypisz zmiennej **x** wartość 15;
- jeśli reszta z dzielenia całkowitego zmiennej **x** (liczby 15) przez 2 jest równa 0, wyświetl połowę tej liczby;
- w przeciwnym wypadku wyświetl liczbę o 1 mniejszą.

```
1. x = 15
2. if x % 2 == 0:
3.     x = x / 2
4. else:
5.     x = x - 1
```

Rys. 1. Przykładowa instrukcja warunkowa złożona z dwóch elementów

Instrukcja warunkowa na rys. 2 składa się z trzech części. Można ją odczytać następująco:

- przypisz zmiennej **x** wartość –12;
- jeśli **x** (liczba –12) jest większa od 0, przypisz zmiennej **zn** 1;
- jeśli poprzedni warunek jest fałszywy i **x** (liczba –12) jest mniejsze od 0, przypisz zmiennej **zn** –1;
- w przeciwnym wypadku przypisz zmiennej **zn** 0.

```
1. x = -12
2. if x > 0:
3.     zn = 1
4. elif x < 0:
5.     zn = -1
6. else:
7.     zn = 0
8. print(zn)
```

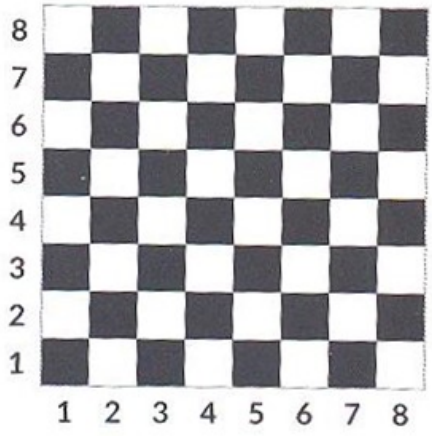
Rys. 2. Przykładowa instrukcja warunkowa złożona z trzech elementów

Ćwiczenie 2. Ile jest białych pól, a ile czarnych

Szachownica złożona jest z kwadratowych pól, naprzemiennie białych i czarnych. Lewy dolny róg (1, 1) zajmuje pole czarne. Przeanalizuj dane w tabeli i zdefiniuj funkcję **ile_kolorowych(n, kolor)**, której parametrami są rozmiar szachownicy i kolor pola – **b** (białe) lub **c** (czarne) – a wynikiem jest liczba pól w danym kolorze. Parametr n możne przyjmować wartości od 2 do 32.

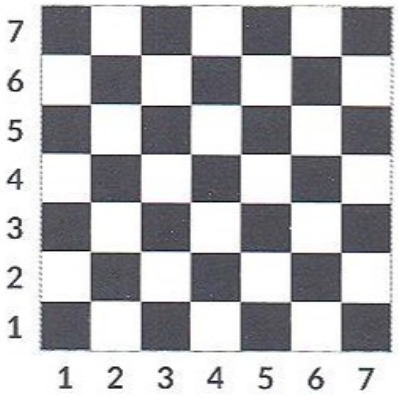
Wywołanie funkcji	Wynik
`ile_kolorowych(8, "b")`	32
`ile_kolorowych(7, "b")`	24
`ile_kolorowych(8, "c")`	32
`ile_kolorowych(7, "c")`	25

- W przypadku wartości parzystych parametru liczba pól w obu kolorach jest jednakowa. Aby wyznaczyć liczbę pól w danym kolorze, musisz pomnożyć przez siebie rozmiar szachownicy i podzielić przez 2: 8 * 8 // 2 = 32.



- W przypadku wartości nieparzystych parametru liczba pól białych jest o 1 mniejsza od liczby pól czarnych. Aby wyznaczyć liczbę pól w danym kolorze, musisz wprowadzić dwa warunki:
  - jeśli pola są białe, wynikiem będzie kwadrat rozmiaru szachownicy, podzielony przez 2: 7 * 7 // 2 = 24;
  - w przeciwnym wypadku (tj. dla pól czarnych) wynikiem będzie kwadrat rozmiaru szachownicy pomniejszony o 1, a następnie podzielony przez 2 i powiększony o 1.

  Powstaje pytanie: dlaczego? Zastanów się i postaraj się uzasadnić podany wzór.



- Zapisz odpowiednie formuły.
  - Liczba pól białych:

```
n * n // 2
```

  - Liczba pól czarnych:

```
(n * n - 1) // 2 + 1
```

- Zdefiniuj funkcję `ile_kolorowych(n, kolor)`.

```
1. def ile_kolorowych(n, kolor):
2.     if kolor == "b":
3.         return n * n // 2
4.     else:
5.         return (n * n - 1) // 2 + 1
```

- Sprawdź działanie skryptu z różnymi danymi, w tym z parzystym i nieparzystym rozmiarem szachownicy.

Jeżeli znasz liczbę pól szachownicy i liczbę pól białych, jak sprawdzić, czy dobrze jest policzona liczba pól czarnych?

PĘTLA FOR

Instrukcja iteracji służy do powtarzania jakiejś instrukcji lub bloku instrukcji w programie, np. dodawania wielu liczb do siebie. Dla każdego elementu sekwencji wykonywany jest podany ciąg instrukcji.

ZAPAMIĘTAJ!

```
for <nazwa zmiennej> in <sekwencja>:
    <instrukcje>
```

W przypadku pętli **for** zwyczajowo używa się zmiennej sterującej pętlą o nazwie **i**, a jako sekwencji iteratora **range(od, do, krok)**. Iterator **range()** może mieć jeden parametr „do" (koniec zakresu), dwa parametry „od" i „do" (początek i koniec zakresu) albo trzy parametry „od", „do" i „krok" (początek zakresu, koniec zakresu i krok, o jaki zwiększa się kolejne wartości), np.

- **range(7)** – definiuje zakres od 0 do 6;
- **range(3, 7)** – definiuje zakres od 3 do 6, bo ostatnia generowana wartość to największa liczba całkowita mniejsza niż ograniczenie górne;
- **range(1, 7, 2)** – definiuje liczby od 1, nie większe niż 7 i co 2, a więc: 1, 3 i 5.

Ćwiczenie 3. Zestawienie liczby pól białych i czarnych

Szachownica złożona jest z kwadratowych pól, naprzemiennie białych i czarnych. Lewy dolny róg (1, 1) zajmuje pole czarne. Efektem wywołania funkcji **testuj_szachownica(n)** powinno być wyświetlenie: kolejnej liczby, łącznej liczby pól szachownicy oraz liczby pól białych i czarnych dla wartości od 2 do 32. Przenalizuj poniższą funkcję, znajdź i popraw błędy.

```
1. def testuj_szachownica(n):
2.     for i in range(2, n):
3.         print (i, ile_pol(i, "b"), ile_kolorowych(i), ile_kolorowych("c", i))
```

Pierwsze wiersze zestawienia będą miały następujące wartości:

n	Liczba pól	Liczba pól białych	Liczba pól czarnych
2	4	2	2
3	9	4	5
4	16	8	8
5	25	12	13
6	36	18	18
7	49	24	25
8	64	32	32
9	81	40	41
10	100	50	50

- Skorzystaj z rozwiązań ćwiczeń 1 i 2. Zwróć uwagę na liczbę i kolejność parametrów. Funkcja **ile_pol(n)** powinna mieć jeden parametr, natomiast funkcja **ile_kolorowych(n, kolor)** – dwa parametry (pierwszy to rozmiar szachownicy, a drugi to kolor szukanych pól).

- Popraw błędy w kodzie funkcji.

```
1. def testuj_szachownica(n):
2.     for i in range(2, 33):
3.         print (i, ile_pol(i), ile_kolorowych(i, "b"), ile_kolorowych(i, "c"))
```

PRZYDATNE FUNKCJE

W języku Python istnieje wiele funkcji, które mogą ułatwić zapis algorytmów. Należą do nich m.in.

- funkcja **min()**, która wyznacza najmniejszą z wartości podanych jako argument – np. wynikiem funkcji **min(5, 2, 11)** jest 2;
- funkcja **max()**, która wyznacza największą z wartości podanych jako argument – np. wynikiem funkcji **max(5, 2, 11)** jest 11.

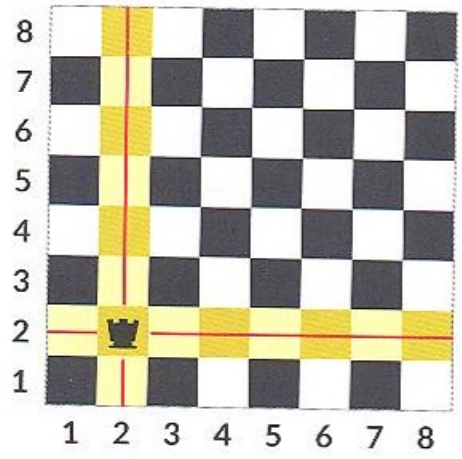
Pełny spis funkcji pomocniczych zawarty jest w dokumentacji języka na stronie **https://docs.python.org/3/library/functions.html**.

Ćwiczenie 4. Ile pól atakuje wieża

Wieża porusza się w tym samym rzędzie lub w tej samej kolumnie, w dowolnym kierunku, o dowolną liczbę pól. Przeanalizuj dane w tabeli i zdefiniuj funkcję **ile_wieza(n, x, y)**, której parametrami są rozmiar szachownicy oraz położenie figury (**x, y**), a wynikiem jest liczba pól atakowanych przez wieżę. Parametr **n** może przyjmować wartości od 2 do 32. Parametry **x** i **y** mogą przyjmować wartości w przedziale od **1** do **n**.

Wywołanie funkcji	Wynik
ile_wieza(8, 2, 2)	**14**
ile_wieza(13, 3, 4)	**24**

- Rozważ liczbę pól atakowanych w pionie i poziomie na szachownicy o rozmiarze 8.



- Zdefiniuj funkcję **ile_wieza(n, x, y)**.

```
1. def ile_wieza(n, x, y):
2.     return 2 * (n - 1)
```

- Sprawdź działanie skryptu z różnymi danymi.

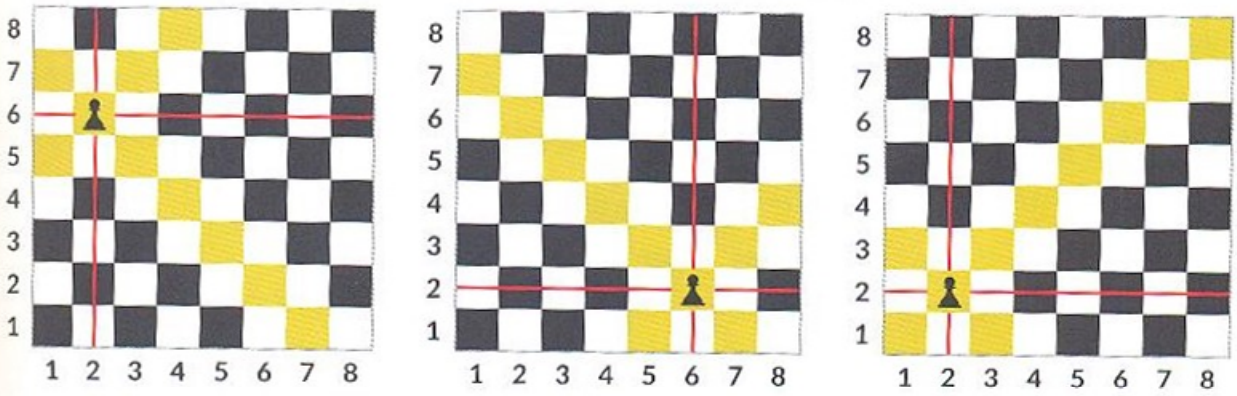
Ćwiczenie 5. Ile pól atakuje goniec

Goniec przesuwa się po przekątnych o dowolną liczbę pól i w dowolnym kierunku. Przeanalizuj dane w tabeli i zdefiniuj funkcję **ile_goniec(n, x, y)**, której parametrami są rozmiar szachownicy oraz położenie figury (**x, y**), a wynikiem jest liczba pól atakowanych przez gońca. Parametr **n** może przyjmować wartości od 2 do 32. Parametry **x** i **y** mogą przyjmować wartości od **1** do **n**.

Wywołanie funkcji	Wynik
ile_goniec(8, 2, 2)	**9**
ile_goniec(13, 3, 4)	**16**

- W sposób naturalny położenie figury dzieli planszę na cztery części. Lewy górny róg, lewy dolny, prawy dolny róg i prawy górny. Rozważmy tę ostatnią część. Liczba pól atakowanych przez gońca o położeniu **(x, y)** to minimum z liczb **n - x** oraz **n - y**. W pozostałych częściach jest podobnie.

Goniec w prawej górnej części atakuje...



2 pola, gdy stoi na pozycji (2, 6) min (8 - 2, 8 - 6) = 2

2 pola, gdy stoi na pozycji (6, 2) min (8 - 6, 8 - 2) = 2

6 pól, gdy stoi na pozycji (2, 2) min (8 - 2, 8 - 2) = 6

- Zapisz odpowiednie formuły.
  - I część:

```
min(x - 1, n - y)
```

  - II część:

```
min(x - 1, y - 1)
```

- III część:

```
min(n - x, y - 1)
```

- IV część:

```
min(n - x, n - y)
```

- Zdefiniuj funkcję **ile_goniec(n, x, y)**.

```
1. def ile_goniec(n, x, y):
2.     return min(x - 1, n - y) + min(x - 1, y - 1) + min(n - x, y - 1)
       + min(n - x, n - y)
```

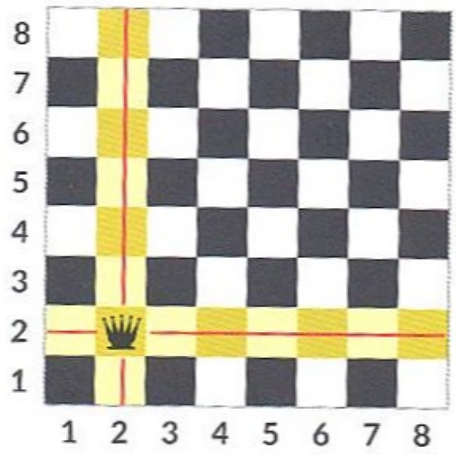
- Sprawdź działanie skryptu z różnymi danymi. Z jakimi danymi warto przetestować funkcję, aby się przekonać, że jest poprawnie napisana?

Ćwiczenie 6. Ile pól atakuje hetman

Przeanalizuj dane w tabeli i zdefiniuj funkcję **ile_hetman(n, x, y)**, której parametrami są rozmiar szachownicy oraz położenie figury (**x, y**), a wynikiem jest liczba pól atakowanych przez hetmana. Parametr **n** może przyjmować wartości od 2 do 32. Parametry **x** i **y** mogą przyjmować wartość w przedziale od **1** do **n**.

Wywołanie funkcji	Wynik
`ile_hetman(8, 2, 2)`	23
`ile_hetman(13, 3, 4)`	40

- Rozważ liczbę pól atakowanych w pionie i poziomie na szachownicy o rozmiarze 8.



- Skorzystaj z rozwiązań ćwiczeń 4 i 5.

- Zdefiniuj funkcję **ile_hetman(n, x, y)**.

```
def ile_hetman(n, x, y):
    return ile_wieza(n, x, y) + ile_goniec(n, x, y)
```

- Sprawdź działanie skryptu z różnymi danymi.

Ćwiczenie 7. Zestawienie liczby pól atakowanych przez hetmana

Dla planszy o rozmiarze 8 wykonaj zestawienie liczby pól atakowanych przez hetmana w zależności od jego położenia.

	1	2	3	4	5	6	7	8
1	21	21	21	21	21	21	21	21
2	21	23	23	23	23	23	23	21
3	21	23	25	25	25	25	23	21
4	21	23	25	27	27	25	23	21
5	21	23	25	27	27	25	23	21
6	21	23	25	25	25	25	23	21
7	21	23	23	23	23	23	23	21
8	21	21	21	21	21	21	21	21

PODSUMOWANIE

- Ważną umiejętnością programistyczną jest zapis w języku formalnym tego, co można wyliczyć bez komputera.
- Podstawy pracy w języku tekstowym obejmują m.in. zapis algorytmu w postaci sekwencji poleceń, zapisywanie obliczeń, wykorzystanie instrukcji warunkowych i pętli oraz definiowanie funkcji.
- Instrukcja warunkowa pozwala na wykonanie różnych obliczeń w zależności od tego, czy zdefiniowane wyrażenie logiczne jest prawdziwe czy fałszywe. Może składać się z jednej, dwóch lub trzech części. Na początku znajduje się słowo kluczowe **if** oraz warunek lub wyrażenie, które mogą mieć wartość **prawda** (1 lub inna wartość) lub **fałsz** (0). Wiersz należy zakończyć dwukropkiem. Potem następuje instrukcja lub ciąg instrukcji do wykonania w przypadku spełnienia warunku. Słowo kluczowe **elif** wprowadza nowy warunek lub wyrażenie oraz instrukcje do wykonania w przypadku spełnienia tego warunku. Słowo kluczowe **else** wprowadza instrukcję do wykonania, gdy żaden z powyższych warunków nie jest prawdziwy.
- Instrukcja iteracji służy do powtarzania jakiejś instrukcji lub bloku instrukcji w programie, np. dodawania wielu liczb do siebie. Dla każdego elementu sekwencji wykonywany jest podany ciąg instrukcji. W przypadku pętli **for** zwykle używa się zmiennej sterującej pętlą o nazwie **i**, a jako sekwencji iteratora **range(od, do, krok)**.

- Funkcja **range()** może mieć jeden parametr „do" (koniec zakresu), dwa parametry „od" i „do" (początek i koniec zakresu) albo trzy parametry „od", „do" i „krok" (początek zakresu, koniec zakresu i krok, o jaki zwiększa się kolejne wartości).
- W języku Python są dostępne standardowe funkcje, choćby takie jak **min()** i **max()**, które odpowiednio wyznaczają najmniejszą lub największą z wartości podanych jako argument.

PYTANIA SPRAWDZAJĄCE

1. Czy w instrukcji warunkowej zawsze występuje słowo kluczowe **else**? Odpowiedź uzasadnij.
2. Jak w języku Python zapisać instrukcję iteracji, by zmienna sterująca pętlą zmieniała wartości od 10 do 0 włącznie?
3. Z jakimi wartościami parametrów warto testować każde zadanie?

ZADANIA DODATKOWE

Zadanie 1. Ile pól atakuje król

Król może się przesuwać pionowo, poziomo i po przekątnej, ale tylko o jedno pole. Przeanalizuj dane w tabeli i zdefiniuj funkcję **ile_krol(n, x, y)**, której parametrami są rozmiar szachownicy oraz położenie figury (**x**, **y**), a wynikiem jest liczba pól atakowanych przez króla. Parametr **n** może przyjmować wartości od 2 do 30. Parametry **x** i **y** mogą przyjmować wartości od **1** do **n**.

Wywołanie funkcji	Wynik
`ile_krol(8, 1, 1)`	3
`ile_krol(10, 4, 4)`	8

Zadanie 2. Ile pól atakuje skoczek

Skoczek porusza się o dwa pola w przód i jedno w bok lub jedno w przód i dwa w bok. Przeanalizuj dane w tabeli i zdefiniuj funkcję **ile_skoczek(n, x, y)**, której parametrami są rozmiar szachownicy oraz położenie figury (**x**, **y**), a wynikiem jest liczba pól atakowanych przez skoczka. Parametr **n** możne przyjmować wartości od 2 do 30. Parametry **x** i **y** mogą przyjmować wartości od **1** do **n**.

Wywołanie funkcji	Wynik
`ile_skoczek(8, 1, 1)`	2
`ile_skoczek(10, 4, 4)`	8

Zadanie 3. Czy można przejść z pola na pole daną figurą

Dostępna jest szachownica o rozmiarze `n` = 8. Przeanalizuj poniższe tabele i zdefiniuj funkcje **`czy_wieza(xp, yp, xk, yk)`**, **`czy_goniec(xp, yp, xk, yk)`** oraz **`czy_hetman(xp, yp, xk, yk)`**, których parametrami są położenie początkowe i końcowe figury, a wynikiem jest **`True`**, jeśli w jednym ruchu można przejść z położenia początkowego do końcowego daną figurą, lub **`False`**, jeśli nie da się tego zrobić. Parametry mogą przyjmować wartości od 1 do 8.

Wywołanie funkcji	Wynik
`czy_wieza(1, 1, 1, 4)`	`True`
`czy_wieza(2, 2, 7, 8)`	`False`

Wywołanie funkcji	Wynik
`czy_goniec(1, 1, 4, 4)`	`True`
`czy_goniec(2, 2, 7, 8)`	`False`

Wywołanie funkcji	Wynik
`czy_hetman(1, 1, 1, 4)`	`True`
`czy_hetman(2, 2, 7, 8)`	`False`