Warunki jakie powinno spełniać rozwiązanie

1. W implementacji można korzystać tylko z elementarnych konstrukcji Python'a, takich jak: funkcje, instrukcje warunkowe, pętle.
2. Nie wolno korzystać z wbudowanych algorytmów sortowania.
3. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych.

Format rozwiązań

Implementacja funkcji powinna się znajdować w pliku o nazwie `zad1.py`. Krótki opis rozwiązania powinien być umieszczony na początku pliku w formie komentarza (w pierwszej linii w komentarzu powinno być imię i nazwisko studenta). Niedopuszczalne jest w szczególności:

1. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
2. wypisywanie na ekranie jakichkolwiek napisów, ew. napisy dodane na potrzeby diagnozowania błędów należy usunąć przed wysłaniem zadania.

Zadania niezgodne z powyższymi ograniczeniami otrzymają ocenę 0 pkt. Rozwiązania w innych formatach (np. `.PDF`, `.DOC`, `.PNG`, `.JPG`) z definicji nie będą sprawdzane i otrzymają ocenę 0 punktów, nawet jeśli będą poprawne.

Czas na rozwiązanie zadania 25 minut.
Oceniane będą: czytelność, poprawność i efektywność rozwiazań.

Zadanie 1

Napis nazywamy wielokrotnym, jeżeli powstał przez n-krotne ($n > 1$) powtórzenie innego napisu o długości co najmniej 1. Przykłady napisów wielokrotnych: $ABCABCABC$, $AAAA$, $ABAABA$. Dana jest tablica $T[N]$ zawierająca napisy. Proszę napisać funkcję `multi(T)`, która zwraca długość najdłuższego napisu wielokrotnego występującego w tablicy T lub wartość 0, jeżeli takiego napisu nie ma w tablicy.

Warunki jakie powinno spełniać rozwiązanie

1. W implementacji można korzystać tylko z elementarnych konstrukcji Python'a, takich jak: funkcje, instrukcje warunkowe, pętle.
2. Należy założyć, że typ int jest ograniczony do liczb 64 bitowych
3. Nie wolno korzystać z typu str.
4. Nie wolno korzystać z wbudowanych algorytmów sortowania.
5. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych.

Format rozwiązań

Implementacja funkcji powinna się znajdować w pliku o nazwie `zad2.py`. Krótki opis rozwiązania powinien być umieszczony na początku pliku w formie komentarza (w pierwszej linii w komentarzu powinno być imię i nazwisko studenta). Niedopuszczalne jest w szczególności:

1. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
2. wypisywanie na ekranie jakichkolwiek napisów, ew. napisy dodane na potrzeby diagnozowania błędów należy usunąć przed wysłaniem zadania.

Zadania niezgodne z powyższymi ograniczeniami otrzymają ocenę 0 pkt. Rozwiązania w innych formatach (np. `.PDF`, `.DOC`, `.PNG`, `.JPG`) z definicji nie będą sprawdzane i otrzymają ocenę 0 punktów, nawet jeśli będą poprawne.

Czas na rozwiązanie zadania 25 minut.
Oceniane będą: czytelność, poprawność i efektywność rozwiazań.

Zadanie 2

Dana jest tablica $T[N][N]$ wypełniona wartościami $0, 1$. Każdy wiersz tablicy traktujemy jako liczbę zapisaną w systemie dwójkowym o długości N bitów. Stała N jest rzędu 1000. Proszę zaimplementować funkcję `distance(T)`, która dla takiej tablicy wyznaczy dwa wiersze, dla których różnica zawartych w wierszach liczb jest największa. Do funkcji należy przekazać tablicę, funkcja powinna zwrócić odległość pomiędzy znalezionymi wierszami. Można założyć, że żadne dwa wiersze nie zawierają identycznego ciągu cyfr.

Warunki jakie powinno spełniać rozwiązanie

1. W implementacji można korzystać tylko z elementarnych konstrukcji Python'a, takich jak: funkcje, rekurencja instrukcje warunkowe, pętle.
2. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych.
3. Niedopuszczalne jest używanie typu str.

Format rozwiązań

Implementacja funkcji powinna się znajdować w pliku o nazwie `zad4.py`. Krótki opis rozwiązania powinien być umieszczony na początku pliku w formie komentarza (w pierwszej linii w komentarzu powinno być imię i nazwisko studenta). Niedopuszczalne jest w szczególności:

1. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
2. wypisywanie na ekranie jakichkolwiek napisów, ew. napisy dodane na potrzeby diagnozowania błędów należy usunąć przed wysłaniem zadania.

Zadania niezgodne z powyższymi ograniczeniami otrzymają ocenę 0 pkt. Rozwiązania w innych formatach (np. `.PDF`, `.DOC`, `.PNG`, `.JPG`) z definicji nie będą sprawdzane i otrzymają ocenę 0 punktów, nawet jeśli będą poprawne.

Czas na rozwiązanie zadania 25 minut.
Oceniane będą: czytelność, poprawność i efektywność rozwiązań.

Zadanie 3

Dana jest tablica $T[N][N]$ (reprezentująca szachownicę) wypełniona liczbami całkowitymi. Proszę zaimplementować funkcję `chess(T)` która ustawia na szachownicy dwie wieże, tak aby suma liczb na „szachowanych" przez wieże polach była największa. Do funkcji należy przekazać tablicę, funkcja powinna zwrócić położenie wież w postaci krotki $(row_1, col_1, row_2, col_2)$.
Uwaga: zakładamy, że pola na których znajdują się wieże nie są szachowane.

Przykładowe wywołania funkcji:

```
print(chess([[4,0,2],[3,0,0],[6,5,3]])) # (0,1,1,0) suma=17
print(chess([[1,1,2,3],[-1,3,-1,4], [4,1,5,4], [5,0,3,6]] # (2,3,3,1) suma=35
```

Warunki jakie powinno spełniać rozwiązanie

1. W implementacji można korzystać tylko z elementarnych konstrukcji Python'a, takich jak: funkcje, rekurencja instrukcje warunkowe, pętle.
2. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych.
3. Niedopuszczalne jest używanie typu str.

Format rozwiązań

Implementacja funkcji powinna się znajdować w pliku o nazwie `zad4.py`. Krótki opis rozwiązania powinien być umieszczony na początku pliku w formie komentarza (w pierwszej linii w komentarzu powinno być imię i nazwisko studenta). Niedopuszczalne jest w szczególności:

1. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
2. wypisywanie na ekranie jakichkolwiek napisów, ew. napisy dodane na potrzeby diagnozowania błędów należy usunąć przed wysłaniem zadania.

Zadania niezgodne z powyższymi ograniczeniami otrzymają ocenę 0 pkt. Rozwiązania w innych formatach (np. `.PDF`, `.DOC`, `.PNG`, `.JPG`) z definicji nie będą sprawdzane i otrzymają ocenę 0 punktów, nawet jeśli będą poprawne.

Czas na rozwiązanie zadania 25 minut.
Oceniane będą: czytelność, poprawność i efektywność rozwiązań.

Zadanie 4

Dana jest liczba naturalna N. Proszę zaimplementować funkcję `divide(N)`, która sprawdza czy jest możliwe pocięcie liczby N na kawałki, tak aby każdy z kawałków był liczba pierwszą oraz liczba kawałków też była liczbą pierwszą. Funkcja powinna zwracać wartość logiczną. Na przykład: `divide(2347)=True`, podział na 23 i 47, natomiast `divide(2255)=False`.

Przykładowe wywołania funkcji:

```
print(divide(273)) # True, podział 2|7|3
print(divide(22222)) # True, podział 2|2|2|2|2
print(divide(23672)) # True, podział 23|67|2
print(divide(2222)) # False
print(divide(21722)) # False
```

Warunki jakie powinno spełniać rozwiązanie

1. W implementacji można korzystać tylko z elementarnych konstrukcji Python'a, takich jak: funkcje, rekurencja instrukcje warunkowe, pętle.

2. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych.

Format rozwiązań

Implementacja funkcji powinna się znajdować w pliku o nazwie `zad5.py`. Krótki opis rozwiązania powinien być umieszczony na początku pliku w formie komentarza (w pierwszej linii w komentarzu powinno być imię i nazwisko studenta). Niedopuszczalne jest w szczególności:

1. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,

2. wypisywanie na ekranie jakichkolwiek napisów, ew. napisy dodane na potrzeby diagnozowania błędów należy usunąć przed wysłaniem zadania.

Zadania niezgodne z powyższymi ograniczeniami otrzymają ocenę 0 pkt. Rozwiązania w innych formatach (np. `.PDF`, `.DOC`, `.PNG`, `.JPG`) z definicji nie będą sprawdzane i otrzymają ocenę 0 punktów, nawet jeśli będą poprawne.

Czas na rozwiązanie zadania 25 minut.
Oceniane będą: czytelność, poprawność i efektywność rozwiązań.

Zadanie 5

Dana jest tablica $T[N]$ wypełniona niepowtarzającymi się liczbami naturalnymi. Proszę zaimplementować funkcję `trojki(T)` która zlicza wszystkie trójki liczb, które spełniają następujące warunki:
(1) największym wspólnym dzielnikiem trzech liczb jest liczba 1,
(2) pomiędzy dwoma kolejnymi elementami trójki może być co najwyżej jedna przerwa.
Funkcja powinna zwrócić liczbę znalezionych trójek.

Przykładowe wywołania funkcji:

```
print(trojki([2,4,6,7,8,10,12])) # 0 trójek
print(trojki([2,3,4,6,7,8,10])) # 1 trójka (3,4,7)
print(trojki([2,4,3,6,5])) # 2 trójki (2,3,5),(4,3,5)
print(trojki([2,3,4,5,6,8,7])) # 5 trójek (2,3,5),(3,4,5),(3,5,8),(5,6,7),(5,8,7)
```

Warunki jakie powinno spełniać rozwiązanie

1. W implementacji można korzystać tylko z elementarnych konstrukcji Python'a, takich jak: funkcje, rekurencja instrukcje warunkowe, pętle.

2. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych.

Format rozwiązań

Implementacja funkcji powinna się znajdować w pliku o nazwie `zad6.py`. Krótki opis rozwiązania powinien być umieszczony na początku pliku w formie komentarza (w pierwszej linii w komentarzu powinno być imię i nazwisko studenta). Niedopuszczalne jest w szczególności:

1. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,

2. wypisywanie na ekranie jakichkolwiek napisów, ew. napisy dodane na potrzeby diagnozowania błędów należy usunąć przed wysłaniem zadania.

Zadania niezgodne z powyższymi ograniczeniami otrzymają ocenę 0 pkt. Rozwiązania w innych formatach (np. `.PDF`, `.DOC`, `.PNG`, `.JPG`) z definicji nie będą sprawdzane i otrzymają ocenę 0 punktów, nawet jeśli będą poprawne.

Czas na rozwiązanie zadania 25 minut.
Oceniane będą: czytelność, poprawność i efektywność rozwiązań.

Zadanie 6

Dana jest definicja klasy, której obiekty stanowią elementy listy odsyłaczowej:

```
class Node
..def init(self,val):
....self.val = val
....self.next = None
```

Zbiór mnogościowy liczb naturalnych reprezentowany jest przez listę o uporządkowanych rosnąco elementach. Proszę napisać funkcję `iloczyn(z1,z2,z3)`, która przekształca 3 listy (zbiory) z1,z2,z3 w jedną listę (zbiór) zawierającą elementy będące częścią wspólna zbiorów z1,z2,z3. Funkcja powinna zwrócić wskazanie do listy wynikowej.

Komentarz: Zadanie jest tak proste, że nie wymaga przykładu ani danych testowych.