GSA
Gdańskie Liceum
Autonomiczne

UCZEŃ:
IMIĘ: .....
NAZWISKO: .....

DIAGNOZA MATURALNA:

INFORMATYKA

POZIOM: ROZSZERZONY

DATA: **6 czerwca 2025 r.**
GODZINA ROZPOCZĘCIA: **7:30**
CZAS TRWANIA: **150 minut**
LICZBA PUNKTÓW DO UZYSKANIA: **30**

ZADANIE 1. PROSTOKĄTNA LICZBA BINARNA

Rozważmy tablicę o ***w*** wierszach i ***k*** kolumnach oraz dodatnią liczbę całkowitą ***n***, której zapis w postaci binarnej ma co najwyżej ***w * k*** cyfr. Tę liczbę zapisujemy w systemie binarnym i wpisujemy otrzymane cyfry w kolejnych komórkach tablicy, począwszy od lewego górnego rogu.
Cyfry zapisu binarnego najpierw wprowadzamy do pierwszego wiersza, następnie – do drugiego, potem – do trzeciego wiersza itd. Jeśli w pewnej komórce zakończymy wprowadzanie ostatniej cyfry zapisu binarnego, to od następnej komórki zaczynamy wprowadzać ponownie cyfry zapisu binarnego tej samej liczby, zaczynając od pierwszej cyfry. Szukamy cyfry znajdującej się w prawym dolnym rogu tablicy.

Przykład. 1.
Weźmy $w = 5$, $k = 3$, $n = 19$.
Przedstawiamy liczbę $n = 19$ w zapisie binarnym: 10011. Wprowadzamy cyfry zapisu binarnego liczby n do tablicy 5 x 3. Zaczynamy od lewego górnego rogu i wpisujemy kolejne cyfry, aż osiągniemy koniec tablicy.

1	0	0
1	1	1
0	0	1
1	1	0
0	1	1

Cyfrą w prawym dolnym rogu jest 1.

ZADANIE 1.1. (0–1)

Wprowadź cyfry zapisu binarnego liczby $n = 179$ do tablicy o wymiarach $w = 4$ i $k = 5$ według powyższej metody.

Miejsce na obliczenia (brudnopis):

.....

.....

.....

.....

ZADANIE 1.2. (0–4)

Dana jest dodatnia liczba całkowita *n*. Cyfry zapisu binarnego liczby *n* wprowadzono w sposób przedstawiony na początku zadania do tablicy o wymiarach $w \times k$.
W pseudokodzie lub języku programowania zapisz algorytm, który wyznaczy cyfrę zapisu binarnego liczby *n* znajdującą się w prawym dolnym rogu tabeli o wymiarach $w \times k$.

Uwaga: W zapisie algorytmu możesz wykorzystać tylko operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie, dzielenie całkowite, reszta z dzielenia), porównywanie liczb, odwoływanie się do pojedynczego elementu tablicy za pomocą jego indeksu, instrukcje sterujące, przypisania do zmiennych lub samodzielnie napisane funkcje, wykorzystujące powyższe operacje. **Zabronione** jest używanie funkcji wbudowanych oraz operatorów innych
niż wymienione.

Specyfikacja

Dane:

- *w* – dodatnia liczba całkowita, liczba wierszy tablicy
- *k* – dodatnia liczba całkowita, liczba kolumn tablicy
- *n* – dodatnia liczba całkowita

Wynik:

- *x* – cyfra w zapisie binarnym liczby *n*, która stoi w dolnym prawym rogu tablicy

Algorytm:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Zadanie 2. Test

Oceń prawdziwość podanych zdań. Zaznacz **P**, jeśli zdanie jest prawdziwe, albo **F** – jeśli jest fałszywe.
W każdym zadaniu punkt uzyskasz tylko za komplet poprawnych odpowiedzi.

Zadanie 2.1. (0–1)

Dany jest algorytm:

$s \leftarrow 0$
dla $i = 1, 2, \ldots, n$
    dla $j = i, i + 1, \ldots, n$
        $s \leftarrow s + 1$

Złożoność obliczeniowa powyższego algorytmu oceniona liczbą wykonań instrukcji $s \leftarrow s + 1$, w zależności od dodatniej liczby całkowitej n, jest

1	liniowa	P	F
2	kwadratowa	P	F
3	n log n	P	F
4	nie większa niż sześcienna	P	F

Zadanie 2.2. (0–1)

Po dodaniu liczb 132_4 oraz 3111_4 zapisanych w systemie czwórkowym otrzymamy:

1	1111011_2	P	F
2	362_8	P	F
3	$F3_{16}$	P	F
4	3303_4	P	F

Zadanie 2.3. (0–1)

W bazie danych istnieje tabela *mandaty*(*numer*, *id_osoby*, *punkty*) zawierająca następujące dane:

numer	Id_osoby	punkty
1	1	5
2	1	14
3	2	20
4	3	21
5	2	1
6	1	2

1	Wynikiem zapytania: **SELECT** id_osoby, sum(punkty) **FROM** mandaty **GROUP BY** id_osoby **HAVING** sum(punkty) > 5 jest zestawienie: 1 14 2 20 3 21	P	F
2	Wynikiem zapytania: **SELECT** id_osoby, sum(punkty) **FROM** mandaty **GROUP BY** id_osoby jest zestawienie: 1 21 2 21 3 21	P	F
3	Wynikiem zapytania: **SELECT** numer + punkty **FROM** mandaty jest 86	P	F
4	Wynikiem zapytania: **SELECT** count(punkty) **FROM** mandaty **WHERE** punkty = 21 jest 1	P	F

Zadanie 3. Liczba Pi

Pewien matematyk jest zafascynowany liczbą π ≈ 3,14159265... do tego stopnia, że zapisał jej rozwinięcie dziesiętne z dokładnością do 10 000 cyfr po przecinku. Wszystkie cyfry po przecinku zapisał w pliku tekstowym `pi.txt`.

Plik `pi.txt` zawiera 10 000 wierszy, każdy wiersz zawiera jedną cyfrę. W pierwszych 10 wierszach pliku zapisano zatem cyfry:
1
4
1
5
9
2
6
5
3
5

Matematyk zastanawia się, jakiego rodzaju regularności można zaobserwować w zebranych danych.

Napisz **program(y)**, który(-e) da(-dzą) odpowiedzi do poniższych zadań. Odpowiedzi do zadań zapisz w pliku `wyniki3.txt`, a każdą z nich poprzedź numerem odpowiedniego zadania.

Plik `pi_przyklad.txt` zawiera 100 pierwszych wierszy pliku `pi.txt`. Odpowiedzi dla danych z tego pliku są podane pod treściami zadań.

Zadanie 3.1. (0–2)

Fragmentem *2-cyfrowym* nazywamy dwie następujące po sobie cyfry w pliku `pi.txt`. Wszystkich fragmentów *2-cyfrowych* zapisanych w tym pliku jest 9 999. Ostatni rozpoczyna się w wierszu nr 9 999.

Przykładowe fragmenty *2-cyfrowe* podano w poniższej tabeli.

i	Fragment 2-cyfrowy złożony z cyfr na pozycjach *i*, *i+1*
1	14
2	41
3	15
9	35

Znajdź liczbę wszystkich fragmentów *2-cyfrowych*, które są zapisami dziesiętnymi liczb o wartościach **większych** od 90.

Dla danych zapisanych w pliku `pi_przyklad.txt` poprawna odpowiedź to 13.

ZADANIE 3.2. (0–3)

Wszystkich możliwych różnych fragmentów *2-cyfrowych* jest dokładnie 100. Są nimi fragmenty 00, 01, 02, …, 99. Można sprawdzić, że np. *2-cyfrowy* fragment równy 27 występuje w pliku `pi.txt` dokładnie 101 razy.

Znajdź fragmenty *2-cyfrowe*, których liczba wystąpień w pliku `pi.txt` jest najmniejsza, oraz fragmenty *2-cyfrowe*, których liczba wystąpień w pliku `pi.txt` jest największa.
W wyniku podaj znalezione fragmenty *2-cyfrowe* oraz liczby ich wystąpień.

W przypadku, gdy więcej niż jeden fragment występuje tyle samo razy, wypisz ten o mniejszej wartości liczbowej.

Dla danych w pliku `pi_przyklad.txt` poprawna odpowiedź to
00 0
62 4

(minimalna liczba wystąpień: fragment 00, liczba wystąpień 0; maksymalna liczba wystąpień: fragment 62, liczba wystąpień 4)

--> Informacja do zadań 3.3. i 3.4. <--

Skończony co najmniej 4-elementowy ciąg liczb ($a_1, a_2, \ldots, a_n$) jest *rosnąco-malejący*, jeśli można podzielić go na dwa ciągi, z których pierwszy jest rosnący, a drugi – malejący, tzn. jeśli istnieje takie $k \in \{2, 3, \ldots, n-2\}$, że $a_1 < a_2 < \ldots < a_k$ oraz $a_{k+1} > a_{k+2} > \ldots > a_n$.

Przykład:
Ciąg (2, 5, 7, 9, 8, 3, 1) jest *rosnąco-malejący*, bo można go podzielić na dwa ciągi: rosnący (2, 5, 7) i malejący (9, 8, 3, 1) lub – odpowiednio – (2, 5, 7, 9) i (8, 3, 1). Ciąg (5, 9, 9, 4,1) także jest *rosnąco-malejący*.
Przykłady ciągów, które nie są *rosnąco-malejące*, to: (2, 5, 8, 4, 3, 4, 5), (1, 2, 3, 4), (5, 5, 3, 2, 1).

ZADANIE 3.3. (0–3)

Podaj, ile jest wszystkich *rosnąco-malejących* ciągów złożonych z dokładnie sześciu kolejnych cyfr zapisanych w pliku `pi.txt`.

Dla pliku `pi_przyklad.txt` poprawna odpowiedź to 3.
(w pliku `pi_przyklad.txt` są trzy ciągi *rosnąco-malejące* złożone z dokładnie sześciu cyfr: 028841, 089986, 899862)

ZADANIE 3.4. (0–2)

Znajdź najdłuższy ciąg kolejnych cyfr z pliku `pi.txt`, który jest *rosnąco-malejący*, oraz pozycję, na której on się rozpoczyna. W pliku `pi.txt` jest tylko jeden taki ciąg o największej długości.

Wynik zapisz w dwóch wierszach: w pierwszym wierszu zapisz pozycję, od której zaczyna się znaleziony ciąg, a w drugim wypisz znaleziony ciąg. Cyfry ciągu zapisz jedną po drugiej, bez znaku odstępu.

Dla danych w pliku `pi_przyklad.txt` poprawna odpowiedź to

77
0899862

(najdłuższy ciąg *rosnąco-malejący* w pliku `pi_przyklad.txt` to ciąg 0899862 o długości 7 rozpoczynający się w 77 wierszu pliku).

Do oceny oddajesz:
- plik tekstowy `wyniki3.txt`, zawierający odpowiedzi do poszczególnych zadań (odpowiedź do każdego zadania powinna być poprzedzona jego numerem)
- plik(i) zawierający(-e) kody źródłowe Twojego(-ich) programu(-ów) o nazwie(-ach) odpowiednio:

zadanie 3.1. ……

zadanie 3.2. ……

zadanie 3.3. ……

zadanie 3.4. ……

ZADANIE 4. TEST OBIEKTOWOŚCI (0-1)

Oceń prawdziwość podanych zdań. Zaznacz **P**, jeśli zdanie jest prawdziwe, albo **F** – jeśli jest fałszywe.

1	Klasa jest konkretną instancją obiektu utworzoną w pamięci komputera	P	F
2	Metoda to funkcja zdefiniowana wewnątrz klasy	P	F
3	Modyfikacja jednego obiektu nie wpływa na inne obiekty tej samej klasy	P	F
4	Ukrywanie danych przed bezpośrednim dostępem z zewnątrz jest złą praktyką programistyczną	P	F

ZADANIE 5. SZACHY

Uwaga: do rozwiązania zadań 5.1. – 5.3. nie jest potrzebna znajomość reguł gry w szachy.

W pliku `szachy.txt` znajduje się zapis partii szachów, jaką w 2020 roku rozegrali polski arcymistrz Jan Krzysztof Duda oraz mistrz świata Magnus Carlssen. Zapis partii składa się z opisów 125 plansz przedstawiających stany gry (położenie bierek na szachownicy) po kolejnych posunięciach każdego z graczy. Opis każdej planszy składa się z:

- 8 wierszy tekstu po 8 znaków w każdym wierszu
- kolejne znaki w wierszach oznaczają:
  - znak '.' – puste pole
  - wielkie litery – białe bierki (czyli białe figury i pionki)
  - małe litery – czarne bierki

– oznaczenia bierek to:
K/k – król, W/w – wieża, S/s – skoczek,
H/h – hetman, G/g – goniec, P/p – pionek.

Dla zachowania czytelności, po każdym opisie następuje pojedynczy pusty wiersz. W dalszej części, zamiast „opis planszy”, będziemy pisać krótko „plansza”.

Przykłady: (na lewo stan gry, na prawo graficzna reprezentacja obrazująca ów stan gry)

```
wsghkgsw
pppppppp
........
........
....P...
........
PPPP.PPP
WSGHKGSW
```

```
wsghkgsw
pp.ppppp
..p.....
........
....P...
........
PPPP.PPP
WSGHKGSW
```

Napisz **program(-y)**, który(-e) znajdzie(-dą) odpowiedzi do poniższych zadań.
Do Twojej dyspozycji jest plik `szachy_przyklad.txt`, który zawiera 9 plansz zapisanych w podanym wyżej formacie. Odpowiedzi dla pliku `szachy_przyklad.txt` podano w treści poszczególnych zadań. Pamiętaj, że Twój(-e) program(-y) musi(-szą) działać dla 125 plansz.

ZADANIE 5.1. (0–3)

Podaj, na ilu planszach znajduje się przynajmniej jedna pusta kolumna, czyli taka, na polach której nie stoi żadna bierka. Podaj także największą liczbę pustych kolumn na jednej z tych plansz.

Odpowiedź dla pliku `szachy_przyklad.txt`:
7 5
(7 plansz z pustymi kolumnami, największa liczba pustych kolumn na planszy – 5).

Do oceny oddajesz:

- plik `zadanie5_1.txt` zawierający odpowiedź do zadania (dwie liczby oddzielone spacją – liczba plansz z pustymi kolumnami oraz największa liczba pustych kolumn na planszy)
- plik(-i) z komputerową realizacją zadania (kodem programu) o nazwie (nazwach):

.....

.....

.....

.....

ZADANIE 5.2. (0–3)

Rozstrzygnij, ile razy w trakcie gry (inaczej: na ilu planszach zapisanych w pliku `szachy.txt`) nastąpiła sytuacja, w której jest równowaga – jest **tyle samo** i **takich samych** czarnych bierek, ile białych. Podaj liczbę takich plansz, a także najmniejszą liczbę bierek (łącznie białych i czarnych) na planszy w stanie równowagi.

Przykład:

```
   A:                B:
.k......          .p......
........          ........
........          ........
....s...          ....s...
....S...          ....S...
........          ........
........          ........
.......K          .......K
```

Plansza A jest w równowadze, a plansza B nie jest w równowadze (czarne i białe nie mają takich samych bierek).

Odpowiedź dla pliku `szachy_przyklad.txt`:
6 4
(6 plansz w równowadze, 4 – najmniejsza liczba bierek na planszy w stanie równowagi)

Do oceny oddajesz:
• plik `zadanie5_2.txt` zawierający odpowiedź do zadania (dwie liczby oddzielone spacją – liczba plansz w stanie równowagi oraz najmniejsza liczba bierek na planszy w stanie równowagi)
• plik(-i) z komputerową realizacją zadania (kodem programu) o nazwie (nazwach)

……

……

ZADANIE 5.3. (0–4)

Wieża szachuje króla przeciwnego gracza, jeśli znajduje się w tym samym wierszu lub w tej samej kolumnie co król i pomiędzy nimi nie ma żadnej innej bierki.
Oblicz i podaj, na ilu planszach biała wieża szachuje czarnego króla oraz na ilu planszach czarna wieża szachuje białego króla.

Odpowiedź dla pliku `szachy_przyklad.txt`:
2 0
(2 razy biała wieża szachuje czarnego króla, 0 razy czarna wieża szachuje białego króla).

Do oceny oddajesz:
• plik `zadanie5_3.txt` zawierający odpowiedź do zadania (dwie liczby oddzielone spacją – liczba plansz, na których biała wieża szachuje czarnego króla, i liczba plansz, na których czarna wieża szachuje białego króla)
• plik(-i) z komputerową realizacją zadania (kodem programu) o nazwie (nazwach)

……

……

ZADANIE 6. NAJWIĘKSZY WSPÓLNY DZIELNIK – NWD (0-1)

Algorytm opisany w Księdze VII *Elementów* Euklidesa pozwala szybko obliczyć największy wspólny dzielnik dwóch liczb naturalnych *a* i *b* – *nwd (a, b)* , z których co najmniej jedna jestwiększa od 0.

Oto rekurencyjny sposób obliczania *nwd(a, b)*:

$$nwd(a,b) = \begin{cases} a \text{ dla } b = 0 \\ nwd(b, a \bmod b) \text{ dla } b \geq 1 \end{cases}$$

gdzie: mod – operator dzielenia modulo; wynikiem jego działania jest **reszta** z dzielenia *a* przez *b*, na przykład 19 mod 7 = 5.

Przykład:

nwd(16,12) = nwd(12, 4) = nwd(4,0) = 4

funkcja *nwd* jest wywoływana w tym przypadku 3 razy:
Podaj liczbę wywołań funkcji dla a=56 i *b*=72 oraz dla a=72 i b=56

……
……
……