WYMAGANIA EDUKACYJNE

PRZEDMIOT: INFORMATYKA

POZIOM: KLASA 2

ZAKRES: ROZSZERZONY

Wymagania na oceny śródroczne (I półrocze) oraz na oceny roczne obejmują wymagania z działów:
I. Arkusz Kalkulacyjny
II. Algorytmy na liczbach całkowitych i tekstach
III. Rozwiązywanie problemów z wykorzystaniem struktur danych

Wymagania na oceny śródroczne (I półrocze) obejmują wymagania z działów od I do II włącznie, zaś na oceny roczne obejmują wszystkie wymagania z działów od I do III włącznie (cały rok szkolny).

1. **Ustala się następujące progi procentowe na oceny bieżące:**
   - niedostateczny 0% – 39%
   - dopuszczający 40% – 50%
   - dostateczny 51% – 70%
   - dobry 71% – 85%
   - bardzo dobry 86% – 95%
   - celujący 96% – 100%

2. **Ocenie podlegają następujące obszary aktywności uczniów:**
   - praca na lekcji,
   - prace pisemne (sprawdziany, kartkówki, zadanie domowe),
   - wypowiedzi ustne (materiał bieżący),
   - praca indywidualna i grupowa,
   - projekt edukacyjny,
   - prezentacje multimedialne,
   - działania pozalekcyjne.

3. **Zakłada się stosowanie następujących form sprawdzania wiadomości i umiejętności uczniów:**
   - sprawdzian,
   - test,
   - kartkówka (zapowiedziana, niezapowiedziana),
   - odpowiedź ustna,
   - praca na lekcji, praca w grupie,
   - zadanie domowe,
   - aktywność na lekcji.

4. **Informacje o sprawdzaniu wiadomości i umiejętności z materiału bieżącego:**
   - materiał bieżący obejmuje trzy ostatnie lekcje,
   - wiadomości i umiejętności sprawdzane są w następujący sposób:
     a. odpowiedź ustna,
     b. odpowiedź pisemna, zadanie praktyczne,
     c. kartkówka – zapowiedziana i niezapowiedziana,
   - można korzystać z „numerków szczęścia",
   - z „numerków szczęścia" nie korzystają uczniowie, którzy posiadają godziny nieusprawiedliwione w poprzednim miesiącu,
   - „numerki szczęścia" nie obowiązują na zapowiedzianej kartkówce,
   - osoby nieobecne podczas kartkówki zapowiedzianej są zobowiązane do jej nadrobienia – termin ustalony z nauczycielem,
   - w przypadku stwierdzenia niesamodzielnego rozwiązywania zadań podczas kartkówki nauczyciel ma prawo unieważnić pracę, co jest równoważne z wystawieniem oceny niedostatecznej.

5. **Informacje o sprawdzianach:**
   - sprawdzian może być poprzedzony jest lekcją powtórzeniową,
   - sprawdzian jest zapowiadany co najmniej tydzień przed terminem, a informacja o nim jest odnotowana w dzienniku lekcyjnym,
   - nie przekłada się sprawdzianów,
   - osoby nieobecne podczas sprawdzianu są zobowiązane do jego uzupełnienia w terminie ustalonym z nauczycielem (wyjątek: osoby, które były przez dłuższy czas nieobecne w szkole [np. 2 tygodnie] i nie zdążyły nadrobić materiału, a uzyskały zgodę nauczyciela uczącego na pisanie w innym terminie),
   - prace pisemne (sprawdziany, testy itp.) oddawane są w terminie do 21 dni,
   - każda ocena ze sprawdzianu może być poprawiana w ustalonym terminie,
   - prace pisemne są przechowywane przez nauczyciela; uczeń i jego rodzice mają możliwość wglądu do prac pisemnych,
   - kartkówki nie podlegają poprawie,

- uczeń jest zobowiązany przygotować się do lekcji z 3 ostatnich tematów,
- uczeń może dwa razy na semestr skorzystać z „nieprzygotowania” i nie być brany pod uwagę przy wyborze do odpowiedzi ustnej,
- uczeń może uzyskiwać za aktywność na lekcji „plusy”. Po zebraniu odpowiedniej liczby plusów uczeń otrzymuje ocenę celującą z aktywności,
- w przypadku stwierdzenia niesamodzielnego rozwiązywania zadań podczas sprawdzianu nauczyciel ma prawo unieważnić pracę, co jest równoważne z wystawieniem oceny niedostatecznej.

6. O podwyższenie przewidywanej rocznej oceny klasyfikacyjnej z zajęć edukacyjnych może ubiegać się uczeń, który:

a. Przystępował do wszystkich przewidzianych przez nauczyciela sprawdzianów i prac pisemnych
b. Skorzystał ze wszystkich oferowanych przez nauczyciela form poprawy

Wymagania edukacyjne niezbędne do otrzymania przez ucznia poszczególnych śródrocznych i rocznych ocen klasyfikacyjnych z informatyki <u>w klasie drugiej</u>

Dopuszczający – 2	Dostateczny – 3	Dobry - 4	Bardzo dobry - 5	Celujący - 6
Arkusz kalkulacyjny				
Uczeń: • wprowadza dane różnego typu do arkusza kalkulacyjnego, • omawia zastosowania korespondencji seryjnej, • wyjaśnia relacje w bazach danych.	Uczeń: • pobiera dane do arkusza kalkulacyjnego ze źródeł zewnętrznych, • filtruje dane w arkuszu kalkulacyjnym, • tworzy różne wykresy w arkuszu kalkulacyjnym w zależności od rodzaju danych, • bierze udział w projektach informatycznych jako członek zespołu.	Uczeń: • przeprowadza analizę danych zgromadzonych w arkuszu kalkulacyjnym, • omawia błąd zaokrąglenia i błąd przybliżenia w obliczeniach komputerowych, • dobiera środowisko informatyczne do rodzaju rozwiązywanego problemu, • wyszukuje informacje zgromadzone w bazach danych, • w bazach danych wykorzystuje kwerendy, filtrowanie	Uczeń: • wykorzystuje zaawansowane formuły, opracowując dane w arkuszu kalkulacyjnym, • stosuje funkcje zaokrąglające liczby, • korzysta z możliwości obliczeń walutowych,	Uczeń: • wykorzystuje narzędzia do prognozowania wyników
Algorytmy na liczbach całkowitych i tekstach				
• definiuje podstawowe pojęcia z algorytmiki i programowania: algorytm, program, warunek, iteracja, rekurencja, • wymienia sposoby reprezentacji algorytmów, • korzysta ze środowiska programistycznego: pisze w nim kod, uruchamia program, odczytuje i zapisuje pliki, • pisze programy o niewielkim stopniu trudności, • omawia pojęcia: złożoność obliczeniowa algorytmu, algorytm naiwny, algorytm optymalny, złożoność pesymistyczna, złożoność oczekiwana (średnia),	• przedstawia krótkie algorytmy w postaci listy kroków, opisu słownego, pseudokodu, • dodaje liczby binarne, • konwertuje liczby między pozycyjnymi systemami liczbowymi, • wykonuje działania arytmetyczne na liczbach w systemach liczbowych o różnych podstawach, • przedstawia liczby w kodzie U2, • definiuje pojęcie zdania logicznego, charakteryzuje podstawowe operacje logiczne (koniunkcja, alternatywa, negacja) oraz operatory logiczne,	• określa specyfikację algorytmu (dane, wynik), • pisze programy o różnym stopniu trudności, szacuje ich efektywność, • przedstawia omawiane algorytmy w postaci opisu słownego, listy kroków, pseudokodu, • dobiera typy danych do realizacji problemu, • pisze programy konwertujące liczby między systemem dziesiętnym i binarnym, • implementuje w języku Python algorytmy wykonujące działania arytmetyczne na liczbach w różnych systemach,	• charakteryzuje sytuacje algorytmiczne, proponuje sposoby ich rozwiązania, • pisze programy o podwyższonym stopniu trudności • optymalizuje rozwiązania, • pisze programy konwertujące liczby między różnymi systemami pozycyjnymi, • w programach wykonujących działania na liczbach w różnych systemach pozycyjnych wykorzystuje typ string i strukturalne typy danych, • wykorzystuje rozwinięcie binarne liczby dziesiętnej w algorytmie szybkiego podnoszenia do potęgi,	• charakteryzuje skomplikowane sytuacje algorytmiczne, proponuje optymalne rozwiązanie sytuacji problemowej z zastosowaniem złożonych struktur danych • pisze programy o wysokim stopniu trudności: z olimpiad przedmiotowych, konkursów informatycznych lub oznaczone trzema gwiazdkami w podręczniku, • wyszukuje palindromy lub anagramy w plikach tekstowych, • tworzy palindromy z napisów, dopisując minimalną liczbę znaków,

• korzysta z podstawowych funkcji języka: operacji wejścia i wyjścia, instrukcji warunkowych i iteracyjnych, gotowych funkcji bibliotecznych, • wymienia podstawowe typy danych, operacje arytmetyczne i logiczne, • definiuje pojęcie systemów liczbowych, • wyjaśnia, czym jest tablica kodów ASCII, • wymienia systemy liczbowe używane w informatyce, • konwertuje liczby między systemami binarnym i decymalnym, • dodaje pisemnie liczby binarne, • wyjaśnia, czym są palindrom i anagram, podaje przykłady, • podaje definicje liczby pierwszej i liczby złożonej, • implementuje w języku Python algorytm zliczający dzielniki danej liczby, • omawia geometryczną interpretację algorytmu Euklidesa,	• pisze programy wykonujące działania na liczbach całkowitych, • korzysta typu string do operacji na łańcuchach znaków, • wykonuje operacje na napisach, wykorzystując odpowiednie metody • tworzy algorytmy sprawdzające, czy napis jest palindromem, • wyjaśnia różnicę między parametrami formalnym i aktualnym, a także między zmiennymi lokalną i globalną, • implementuje w języku Python algorytm naiwny sprawdzający, czy liczba jest pierwsza, • implementuje w języku Python algorytm Euklidesa w wersjach z dzieleniem i odejmowaniem,	• w algorytmach zamiany wykorzystuje zależności między systemami binarnym, ósemkowym i heksadecymalnym, • omawia sposób reprezentacji obrazów w komputerze, korzystając z takich pojęć jak: piksel, model RGB, kanał alfa, • wyjaśnia, na czym polega digitalizacja(dyskretyzacja) dźwięku, • wyjaśnia zasadę tworzenia animacji, • implementuje w języku Python algorytmy sprawdzające, czy napis jest palindromem, • implementuje w języku Python i optymalizuje algorytm sprawdzający, czy liczba jest pierwsza, • pisze program rozkładający liczby na czynniki pierwsze, • stosuje w programach algorytm Euklidesa do obliczenia NWD i NWW, • wykorzystuje algorytm Euklidesa do działań na ułamkach, • szyfruje dane wczytane z pliku tekstowego, • zapisuje w postaci programu rozszerzony algorytm Euklidesa, wyjaśnia jego działanie i zastosowanie,	• wykonuje operacje arytmetyczne na liczbach w różnych systemach, implementuje je w języku Python, • stosuje różne sposoby przekazywania parametrów do funkcji, uzasadnia ich użycie, • pisze funkcje typu logicznego, np. sprawdzającą, czy napis jest palindromem, • szyfruje dane wczytane z pliku z uwzględnieniem polskich znaków diakrytycznych, • pisze program odczytujący informację ukrytą za pomocą szyfru Cezara z wykorzystaniem analizy częstości znaków w tekście, • wykorzystuje algorytm Euklidesa do działań na ułamkach, stosując struktury lub pary	• pisze program rozkładający liczbę złożoną na dwie liczby pierwsze (hipoteza Goldbacha), • implementuje w języku Python algorytm Euklidesa, stosując iterację i rekurencję, • pisze programy szyfrujące i deszyfrujące z wykorzystaniem zaawansowanych szyfrów (np. permutacyjny lub Vigenere'a) i różnych kluczy,
Rozwiązywanie problemów z wykorzystaniem struktur danych				
• w pisanych programach korzysta ze strukturalnych typów danych: napisów, struktur, list, • definiuje pojęcia: kryptologia, kryptografia,kryptoanaliza, tekst jawny, klucz, szyfrogram,	• przedstawia w postaci algorytmu problem wyszukiwania anagramów, • przy pisaniu programów stosuje własne funkcje	• stosuje różne sposoby przekazywania parametrów do funkcji: przez wartość, referencję lub wskaźnik,	• dobiera struktury danych i metody do rodzaju problemu, • sprawdza, czy napisy są anagramami, stosując sortowanie lub zliczanie znaków,	• stosuje w programach algorytmy sortowania inne niż omawiane na lekcjach (np. heapsort), • w projektach zespołowych przyjmuje rolę lidera

• rozróżnia szyfry podstawieniowe i przestawieniowe, • omawia szyfr Cezara jako przykład szyfru podstawieniowego • wyjaśnia, na czym polega łamanie szyfru, • omawia algorytm zliczania znaków w tekście,	• pisze program szyfrujący napis szyfrem Cezara, • omawia algorytm sita Eratostenesa, • przedstawia algorytmy znajdowania spójnych podciągów, wyznaczania najdłuższego z nich oraz podciągu o największej sumie elementów	• pisze programy sprawdzające czy dwa napisy są anagramami, wykorzystując funkcję sort z • stosuje algorytm wyszukiwania binarnego i oszacowuje jego złożoność czasową, • pisze programy sortujące metodami prostymi z zastosowaniem funkcji • pisze program realizujący algorytm sita Eratostenesa,	• przy testowaniu liczby na pierwszość stosuje funkcję typu logicznego, • wyszukuje liczby bliźniacze,	