WYMAGANIA EDUKACYJNE

PRZEDMIOT: INFORMATYKA

POZIOM: KLASA 3

ZAKRES: ROZSZERZONY

Wymagania na oceny śródroczne (I półrocze) oraz na oceny roczne obejmują wymagania z działów:
I. Rozwiązywanie problemów z wykorzystaniem struktur danych
II. Metody algorytmiczne
III. Rozwiązywanie problemów z wykorzystaniem dynamicznych 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 trzeciej</u>

Dopuszczający – 2	Dostateczny – 3	Dobry - 4	Bardzo dobry - 5	Celujący - 6
Rozwiązywanie problemów z wykorzystaniem struktur danych				
Uczeń: • wypisuje liczby pierwsze z zadanego przedziału, stosując metodę sita Eratostenesa, • wyszukuje w ciągu liczb spójne podciągi (nierosnący, niemalejący, stały), wskazuje najdłuższe, oblicza ich sumę,	Uczeń: • omawia algorytm zliczania znaków w tekście oraz wyszukujący maksimum z wykorzystaniem tablic, • przedstawia w postaci listy kroków algorytmy sortowania prostego (bąbelkowe, przez wybieranie) oraz szybkiego i przez scalanie, określa operacje dominujące, • implementuje w języku Python algorytmy rekurencyjne: obliczanie elementów ciągu Fibonacciego, wartości silni i potęgi, • omawia rozszerzony algorytm Euklidesa, • formułuje algorytm wydawania reszty minimalną liczbą monet, znajdowania drogi metodami zachłanną i dynamiczną,	Uczeń: • implementuje w języku Python algorytm zliczania znaków w tekście oraz wyszukujący maksimum z wykorzystaniem tablic, • implementuje w języku Python algorytmy wyszukujące spójne podciągi o różnych cechach,	Uczeń: • stosuje zaawansowane funkcje środowiska i języka programowania (np. z dodatkowych modułów) • optymalizuje program realizujący algorytm sita Eratostenesa i szacuje jego złożoność czasową, • wyszukuje spójne podciągi w plikach tekstowych, stosując optymalne algorytmy (w tym programowanie dynamiczne), wyjaśnia ich działanie, • pisze programy wyszukujące lidera i idola w zbiorze, optymalizuje je, szacuje złożoność czasową,	Uczeń: • stosuje zaawansowane algorytmy i struktury danych do wyszukiwania spójnych podciągów,
Metody algorytmiczne				
• wyjaśnia, na czym polega metoda „dziel i zwyciężaj”, • wczytuje dane z pliku tekstowego, zapisuje wyniki w pliku, • omawia algorytmy wyszukiwania liczby w zbiorach	• implementuje w języku Python algorytmy wyszukiwania liniowego i liniowego z wartownikiem, porównuje ich efektywność,	• stosuje algorytmy sortowania szybkiego i przez scalanie, • stosuje metodę zachłanną w programach – problem kasjera, wyszukiwanie drogi, • porównuje algorytmy iteracyjne i rekurencyjne (liczbę	• pisze program wyszukujący jednocześnie minimum i maksimum w zbiorze z wykorzystaniem metody „dziel i zwyciężaj” oraz podaje wzór na liczbę wykonywanych operacji,	• implementuje w języku Python algorytm wyszukiwania binarnego w wersji rekurencyjnej, • pisze programy sortujące dane różnego typu w plikach tekstowych (liczby, napisy, pary),

uporządkowanym i nieuporządkowanym, • stosuje funkcję losującą w tworzonych programach, • omawia metody sortowania prostego (bąbelkowe, przez wybieranie) oraz szybkiego i przez scalanie na przykładowych danych, • definiuje pojęcia iteracji i rekurencji, • omawia zasadę złotego podziału, • opisuje rozszerzony algorytm Euklidesa, • omawia metody zachłanne na przykładzie problemu kasjera, harmonogramu sali, porównuje metody zachłanną i dynamiczną,		wykonywanych operacji), szacuje ich złożoność czasową,	• szacuje złożoność obliczeniową programów sortujących, modyfikuje funkcje sortujące, zmieniając porządek sortowania, • wykorzystuje poznane algorytmy do rozwiązywania problemów nieomawianych na lekcjach, • pisze programy obliczające liczbę operacji przenoszenia krążków w problemie wież Hanoi, stosując iterację i rekurencję, • do implementacji rozszerzonego algorytmu Euklidesa stosuje zarówno iterację, jak i rekurencję, • stosuje metody zachłanną i dynamiczną w problemach kasjera i wyszukiwania drogi, wskazuje wady i zalety obu metod, szacuje złożoność czasową,	• stosuje zaawansowane algorytmy wyszukiwania, np. najlepszego wyboru (trwałych par), stosując rekurencję, • pisze programy obliczające liczbę operacji przenoszenia krążków w problemie wież Hanoi, stosując iterację i rekurencję,
Rozwiązywanie problemów z wykorzystaniem dynamicznych struktur danych				
• pisze programy o niewielkim stopniu trudności, • wyjaśnia, co to jest notacja infiksowa, notacja prefiksowa, odwrotna notacja polska, drzewo wyrażenia algebraicznego, • definiuje pojęcie dynamicznej struktury danych, • definiuje dynamiczne struktury danych takie jak: stos, kolejka, lista • definiuje graf, wymienia elementy i rodzaje grafów, wymienia sposoby reprezentacji grafu (macierz sąsiedztwa, lista sąsiedztwa),	• wyróżnia operacje, które można wykonywać na dynamicznych strukturach danych (stosie, kolejce, liście), • omawia zastosowanie dynamicznych struktur danych na różnych przykładach, • zapisuje wyrażenia algebraiczne bez użycia nawiasów, w tym w postaci odwrotnej notacji polskiej, • oblicza wartość wyrażenia arytmetycznego zapisanego w odwrotnej notacji polskiej, • omawia algorytmy znajdowania wyjścia z labiryntu z wykorzystaniem iteracji i rekurencji,	• pisze programy o różnym stopniu trudności, szacuje ich efektywność, • dobiera typy danych do rozwiązania problemu, • do przeglądania grafu stosuje algorytm przeszukiwania w głąb (DFS) oraz algorytm przeszukiwania grafu wszerz (BFS), • omawia algorytm Dijkstry,	• charakteryzuje sytuacje algorytmiczne, proponuje sposoby ich rozwiązania, • pisze programy o podwyższonym stopniu trudności: rozwiązuje zadania oznaczone trzema gwiazdkami w podręczniku, • optymalizuje rozwiązania, • stosuje zaawansowane funkcje środowiska i języka programowania, • dobiera struktury danych i metody do rodzaju problemu, • szacuje złożoność algorytmów, • implementuje algorytmy grafowe – BFS, DFS, algorytm Dijkstry, • w reprezentacji liczb rzeczywistych w komputerze	• 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, optymalizuje programy, szacuje ich efektywność • wykorzystuje poznane algorytmy do rozwiązywania problemów nieomawianych na lekcjach, np. sprawdzanie spójności grafu,

	• omawia algorytm przeszukiwania grafu w głąb (DFS), • omawia algorytm przeszukiwania grafu wszerz (BFS), • wyjaśnia, do czego służy algorytm Dijkstry, • wyjaśnia różnicę między przekazywaniem parametrów do funkcji przez wartość i przez referencję,		stosuje reprezentację stało- lub zmiennoprzecinkową zgodnie ze specyfikacją algorytmu, minimalizując błędy w obliczeniach,	