WYMAGANIA EDUKACYJNE NA POSZCZEGÓLNE OCENY

Informatyka

LO 4-letnie

Klasa III (rozszerzona) – I półrocze

Dział / ocena	Opis wymagań
1. Rozwiązywanie problemów z wykorzystaniem dynamicznych struktur danych	
dopuszczający	Uczeń potrafi: • 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, vector, • wymienia rodzaje list, • wyjaśnia, na czym polega sortowanie leksykograficzne, • definiuje graf, wymienia elementy i rodzaje grafów, wymienia sposoby reprezentacji grafu (macierz sąsiedztwa, lista sąsiedztwa),
dostateczny	• wyróżnia operacje, które można wykonywać na dynamicznych strukturach danych (stosie, kolejce, liście, typie vector), • 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, • symuluje problem Flawiusza, • sortuje dane leksykograficznie, • stosuje typ `vector` do reprezentacji grafu w postaci list sąsiedztwa, • 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, • wykorzystuje pliki tekstowe do wczytywania danych i zapisywania wyników,

dobry	• 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,
bardzo dobry	• 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,
celujący	• charakteryzuje skomplikowane sytuacje algorytmiczne, • proponuje optymalne rozwiązanie sytuacji problemowej • stosuje złożone struktur danych,
2. Rozwiązywanie problemów z wykorzystaniem dynamicznych struktur danych	
dopuszczający	• omawia różnice między stałoprzecinkową a zmiennoprzecinkową reprezentacją liczb rzeczywistych w komputerze, • wymienia rodzaje błędów w obliczeniach komputerowych, rozróżnia błąd względny i bezwzględny, • znajduje wartość wielomianu algorytmem naiwnym, • wie, na czym polegają podstawowe metody obliczeń przybliżonych, • zna proste algorytmy badające własności geometryczne (np. położenie punktu względem prostej), • wyjaśnia, co to jest fraktal, wskazuje przykłady struktur fraktalnych występujących w przyrodzie,
dostateczny	• omawia algorytm znajdujący rozwinięcie binarne nieskracalnego ułamka właściwego, • zapisuje liczby w postaci znormalizowanej, • definiuje liczby pojedynczej precyzji i liczby podwójnej precyzji,

	• wykonuje działania na liczbach zmiennoprzecinkowych, • wskazuje różnice między algorytmem stabilnym a algorytmem niestabilnym, • znajduje pierwiastki równania kwadratowego algorytmem stabilnym i algorytmem niestabilnym, • implementuje algorytm obliczający wartość wielomianu z zastosowaniem schematu Hornera, • stosuje w algorytmach numerycznych metody: bisekcji, Newtona–Raphsona, trapezów, prostokątów, • omawia algorytmy badające własności geometryczne – położenie punktu względem prostej, przecinania się odcinków, przynależności punktu do figury, • podaje przykłady fraktali (zbiór Cantora, drzewo binarne, dywan Sierpińskiego, płatek Kocha), wyjaśnia sposób tworzenia tych fraktali,
dobry	• znajduje reprezentację liczby zapisanej w systemie dziesiętnym jako liczby pojedynczej i liczby podwójnej precyzji, • świadomie używa typów `float` i `double` w zadaniach, • stosuje schemat Hornera do zamiany liczby w systemie pozycyjnym o wybranej podstawie na liczbę dziesiętną, • stosuje metodę Monte Carlo w obliczeniach przybliżonych, • w algorytmach badających własności geometryczne wykorzystuje macierz oraz regułę Sarrusa do obliczania wyznacznika macierzy,
bardzo dobry	• w reprezentacji liczb rzeczywistych w komputerze stosuje reprezentację stało- lub zmiennoprzecinkową zgodnie ze specyfikacją algorytmu, minimalizując błędy w obliczeniach, • stosuje schemat Hornera do szybkiego podnoszenia do potęgi, • implementuje algorytmy numeryczne: znajdowania miejsc zerowych funkcji oraz obliczania pierwiastka kwadratowego metodą bisekcji, obliczania pierwiastka kwadratowego metodą Newtona–Raphsona, obliczania pola obszaru zamkniętego metodą prostokątów i metodą trapezów, znajdowania przybliżenia liczby pi oraz symulacja ruchów Browna metodą Monte Carlo, • implementuje algorytmy badające własności geometryczne, • implementuje w języku JavaScript algorytmy generujące fraktale danego stopnia, • stosuje metodę IFS do tworzenia fraktali w arkuszu kalkulacyjnym,

celujący	• optymalizuje programy, szacuje ich efektywność, • wykorzystuje poznane algorytmy do rozwiązywania złożonych problemów obliczeniowych
3. Zaawansowane algorytmy i techniki programistyczne	
dopuszczający	• wyszukuje wzorzec w tekście algorytmem naiwnym, • rozumie działanie funkcji haszującej, • wskazuje różnice między kryptografią symetryczną i kryptografią asymetryczną, definiuje pojęcia klucz publiczny i klucz prywatny, • wyjaśnia, do czego służy algorytm RSA, i wyróżnia główne etapy tego algorytmu (generowanie kluczy, szyfrowanie z kluczem publicznym oraz deszyfrowanie z kluczem prywatnym), • definiuje programowanie strukturalne, • definiuje programowanie obiektowe i podstawowe pojęcia z nim związane,
dostateczny	• implementuje algorytm naiwny wyszukiwania wzorca w tekście, • wyjaśnia metodę haszowania, • wyjaśnia, jak generuje się klucze publiczny i prywatny oraz szyfruje i deszyfruje informacje w algorytmie RSA, • wyjaśnia, na czym polegają metoda zstępująca i metoda wstępująca, • w programowaniu obiektowym definiuje własne klasy, korzystając ze specyfikatorów dostępu,
dobry	• omawia algorytm Karpa–Rabina do wyszukiwania wzorca w tekście z zastosowaniem funkcji haszującej, • w programowaniu obiektowym stosuje hierarchię klas, wyjaśnia, na czym polega hermetyzacja danych i jakie jest zastosowanie operatora zasięgu,
bardzo dobry	• stosuje programowanie obiektowe, definiując własne klasy, obiekty, atrybuty i metody, deklaruje konstruktory w klasach, wyjaśnia, na czym polega polimorfizm i czym są metody wirtualne, • pisze program generujący klucz prywatny i klucz publiczny w algorytmie RSA,
celujący	• stosuje funkcję haszującą oraz algorytm Karpa–Rabina w programach wyszukujących wzorzec w tekście,

	• pisze programy szyfrujące i deszyfrujące informacje w algorytmie RSA,

II półrocze

4. Relacyjne bazy danych	
dopuszczający	• zna podstawowe pojęcia dotyczące relacyjnych baz danych, • wie, co to jest język SQL, zna podstawowe klauzule tego języka, • zna zasady tworzenia zapytań do bazy z wykorzystaniem języka SQL, • wyróżnia etapy pracy nad aplikacją internetową, rozróżnia technologie back-end i front-end,
dostateczny	• wyszukuje informacje w bazach danych, tworzy formularze, kwerendy i raporty, • wykorzystuje język SQL do tworzenia i usuwania baz danych, dodawania tabel do baz danych, usuwania tabel z baz, dodawania rekordów do tabel, importowania danych do tabel, edycji rekordów, • tworzy konta użytkowników i przydziela im uprawnienia do wybranej bazy, używając języka SQL, • formułuje zapytania zwracające określone dane, sortuje wyniki zapytań, • wyjaśnia, na czym polega praca nad aplikacją internetową, instaluje i konfiguruje niezbędne oprogramowanie, przygotowuje bazę danych na potrzeby projektu,
dobry	• projektuje i tworzy proste bazy danych, • przy tworzeniu aplikacji internetowej projektuje i tworzy interfejs użytkownika, zapewnia komunikację aplikacji z bazą danych,
bardzo dobry	• projektuje zaawansowane relacyjne bazy danych, zarządza nimi, tworzy tabele pomostowe, formularze, kwerendy i raporty, • formułuje zapytania w języku SQL, stosując selekcję, sortowanie, projekcję oraz agregowanie danych,
celujący	• tworzy aplikacje internetowe z przejrzystym interfejsem użytkownika korzystające z sieciowej bazy danych, testuje je i wprowadza poprawki, • projektuje rozbudowane relacyjne bazy danych, zarządza nimi, wykorzystując zaawansowane narzędzia oraz klauzule języka SQL,