KLUCZ ODPOWIEDZI · DLA NAUCZYCIELA

Sprawdzian ze struktur danych

Grupa A

1. (1 pkt) **C**

2. (1 pkt) **D**

3. (3 pkt) **1. F; 2. P; 3. P** — po 1 pkt za zdanie

4. (4 pkt) **1–D, 2–B, 3–A, 4–C** — po 1 pkt za parę

5. (4 pkt) **1–B, 2–D, 3–C, 4–A** — po 1 pkt za parę

6. (2 pkt) **(1) kolejka; (2) push** — po 1 pkt za lukę

7. (2 pkt) Przykładowa odpowiedź: Push 3 → S=[3] Push 7 → S=[3,7] Push 1 → S=[3,7,1] Pop → zwraca 1, S=[3,7] Push 5 → S=[3,7,5] Pop → zwraca 5, S=[3,7] Pop → zwraca 7, S=[3] Pobrane wartości: 1, 5, 7 Pozostaje na stosie: 3
Punktacja: Poprawne wartości pobrane przez pop (1 pkt). Poprawny stan końcowy stosu (1 pkt).

8. (3 pkt) Przykładowa odpowiedź: Tablica przechowuje elementy w ciągłym obszarze pamięci, dzięki czemu dostęp do elementu o dowolnym indeksie ma złożoność O(1) — obliczany jest bezpośredni adres w pamięci. W liście jednokierunkowej, aby dotrzeć do n-tego elementu, trzeba przejść kolejno przez wszystkie poprzedzające węzły, co daje złożoność O(n). Z kolei wstawienie nowego elementu na początek listy jednokierunkowej wymaga jedynie zmiany jednego wskaźnika — O(1), natomiast w tablicy statycznej trzeba przesunąć wszystkie istniejące elementy o jedną pozycję — O(n). Tablica jest więc korzystniejsza przy częstym dostępie po indeksie, a lista — przy częstym wstawianiu na początek.
Punktacja: Poprawne wskazanie, że dostęp do elementu tablicy po indeksie ma złożoność O(1), a w liście O(n) (1 pkt) + poprawne wskazanie, że wstawienie na początek listy ma złożoność O(1), a w tablicy statycznej wymaga przesunięcia wszystkich elementów O(n) (1 pkt) + poprawny wniosek/podsumowanie, kiedy która struktura jest korzystniejsza (1 pkt). Razem 3 pkt.

9. (3 pkt) Przykładowa odpowiedź: Drzewo BST (binary search tree) to drzewo binarne, w którym dla każdego węzła wszystkie wartości znajdujące się w jego lewym poddrzewie są mniejsze od wartości tego węzła, a wszystkie wartości w prawym poddrzewie są od niej większe. Dzięki tej własności wyszukiwanie elementu polega na porównywaniu go z kolejnymi węzłami i schodzeniu w lewo lub w prawo, co przy zrównoważonym drzewie o wysokości $\log_2 n$ daje złożoność czasową $O(\log n)$.
Punktacja: Poprawna definicja BST jako drzewa binarnego przechowującego wartości (1 pkt) + poprawne sformułowanie własności porządkującej: wszystkie wartości w lewym poddrzewie mniejsze, w prawym większe od wartości węzła (1 pkt) + poprawna złożoność $O(\log n)$ dla drzewa zrównoważonego wraz z uzasadnieniem (1 pkt). Maksymalnie 3 pkt.

10. (3 pkt) Przykładowa odpowiedź: Przechodzimy przez wyrażenie znak po znaku. Gdy napotkamy nawias otwierający ((, [, {), odkładamy go na stos. Gdy napotkamy nawias zamykający, zdejmujemy element ze szczytu stosu i sprawdzamy, czy jest to nawias otwierający tego samego typu — jeśli nie (lub stos jest pusty), wyrażenie jest niepoprawne. Po przetworzeniu całego wyrażenia sprawdzamy, czy stos jest pusty — jeśli tak, wszystkie nawiasy zostały poprawnie domknięte i wyrażenie jest zbalansowane. Dla {[a+(b*c)]-d}: kolejno odkładane są {, [, (, a przy napotkaniu),], } zdejmowane są w odwrotnej kolejności, aż stos jest pusty — wyrażenie jest poprawne.
Punktacja: Poprawny opis zasady: napotkanie nawiasu otwierającego powoduje odłożenie go na stos (1 pkt) + poprawny opis: napotkanie nawiasu zamykającego powoduje zdjęcie elementu ze szczytu stosu i sprawdzenie, czy pasuje typem do zamykanego nawiasu (1 pkt) + poprawny warunek końcowy: wyrażenie jest zbalansowane, gdy po przetworzeniu całego napisu stos jest pusty (1 pkt). Razem 3 pkt.

Grupa B

1. (1 pkt) **B**

2. (1 pkt) **C**

3. (3 pkt) **1. F; 2. P; 3. F** — po 1 pkt za zdanie

4. (4 pkt) **1–A, 2–C, 3–B, 4–D** — po 1 pkt za parę

5. (4 pkt) **1–C, 2–D, 3–A, 4–B** — po 1 pkt za parę

6. (2 pkt) **(1) kolejką; (2) push** — po 1 pkt za lukę

7. (2 pkt) Przykładowa odpowiedź: Stos działa zgodnie z zasadą LIFO (Last In, First Out) — element dodany do stosu jako ostatni (operacją `push`) jest zawsze pierwszym elementem usuwanym (operacją `pop`); dostęp do pozostałych elementów bez usunięcia elementów znajdujących się nad nimi nie jest możliwy. Przykładem zastosowania stosu jest zarządzanie stosem wywołań funkcji w programie — gdy funkcja wywołuje inną funkcję, informacja o miejscu powrotu jest odkładana na stos, a po zakończeniu wywołanej funkcji zdejmowana z jego wierzchołka.
Punktacja: Poprawne wyjaśnienie zasady LIFO (Last In, First Out) — ostatni dodany element jest pierwszym usuwanym (1 pkt). Podanie trafnego przykładu zastosowania, np. obsługa wywołań funkcji (stos wywołań), historia przeglądarki (cofanie), sprawdzanie poprawności nawiasów (1 pkt). Łącznie 2 pkt.

8. (3 pkt) Przykładowa odpowiedź: Extract-min z min-heap: (1) Pobieramy korzeń (indeks 1) — to minimum. (2) Przenosimy ostatni element tablicy na pozycję korzenia i zmniejszamy rozmiar kopca o 1. (3) Uruchamiamy sift-down od korzenia: porównujemy bieżący węzeł z lewym (indeks 2i) i prawym dzieckiem (2i+1); jeśli dziecko jest mniejsze od rodzica, zamieniamy je z mniejszym z dzieci i kontynuujemy rekurencyjnie w dół. Proces trwa najdłużej przez wysokość drzewa h = $\lfloor \log_2 n \rfloor$, więc złożoność to O(log n).
Punktacja: 1. Krok 1: zamiana korzenia (minimum) z ostatnim elementem tablicy, zmniejszenie rozmiaru kopca o 1 (1 pkt). 2. Krok 2: uruchomienie procedury sift-down (min-heapify) od korzenia — porównanie z dziećmi i zamiana z mniejszym dzieckiem, rekurencyjnie do liścia lub do spełnienia własności kopca (1 pkt). 3. Złożoność O(log n) wynikająca z wysokości drzewa $\lfloor \log_2 n \rfloor$ (1 pkt).

9. (3 pkt) Przykładowa odpowiedź: Min-heap: każdy rodzic jest ≤ swoich dzieci. Insert(3): dodaj 3 na końcu → [1,5,4,8,9,3]. Sift-up: 3 < parent(4) na indeksie 2 → swap → [1,5,3,8,9,4]. Teraz parent(3)=1 ≤ 3 — koniec. Wynik: [1,5,3,8,9,4]. Extract-min: zdejmij 1, wstaw 4 na korzeń → [4,5,3,8,9]; sift-down: 4 > min(5,3)=3 → swap → [3,5,4,8,9].
Punktacja: Definicja min-heap: rodzic ≤ dzieci, drzewo kompletne (1 pkt). Insert: dodaj element na końcu, sift-up do momentu spełnienia własności kopca — $O(\log n)$ (1 pkt). Extract-min: zdejmij korzeń, wstaw ostatni element na korzeń, sift-down — $O(\log n)$; ilustracja: [1,5,4,8,9,3], 3<4 → swap, [1,5,3,8,9,4] (1 pkt). Razem: 3 pkt.

10. (3 pkt) Przykładowa odpowiedź: Macierz sąsiedztwa: pamięć $O(V^2)$, sprawdzenie krawędzi $O(1)$, sąsiedzi $O(V)$. Lista sąsiedztwa: pamięć $O(V + E)$, sprawdzenie krawędzi $O(\deg v)$, sąsiedzi $O(\deg v)$. Macierz lepsza dla gęstych grafów, lista dla rzadkich.
Punktacja: Macierz sąsiedztwa: $O(V^2)$, $O(1)$, $O(V)$ (1 pkt). Lista sąsiedztwa: $O(V + E)$, $O(\deg v)$, $O(\deg v)$ (1 pkt). Rekomendacja: macierz dla grafów gęstych ($E \approx V^2$), lista dla rzadkich ($E \ll V^2$) (1 pkt).

Skala ocen (orientacyjna, 26 pkt w każdej grupie)

bardzo dobry (5)	24–26 pkt
dobry (4)	20–23 pkt
dostateczny (3)	13–19 pkt
dopuszczający (2)	8–12 pkt
niedostateczny (1)	0–7 pkt

Progi: 90%, 75%, 50%, 30%. Progi w Twojej szkole mogą być inne. W zadaniach otwartych uznaj każdą odpowiedź poprawną merytorycznie, także sformułowaną inaczej niż przykład.

Ten sprawdzian online, z wyjaśnieniem każdej odpowiedzi: **matury-online.pl/sprawdzian/informatyka/struktury-danych**. Nauczyciele i korepetytorzy ułożą własną kartę z banku ponad 20 tys. zadań: **matury-online.pl/dla-korepetytorow**.