

Programowanie strukturalne (2024) - Egzamin (przykładowy) - Zestaw S10

Zasady egzaminu:

- **Zadanie 1: 6 pkt. Zadanie 2: 12 pkt. Zadanie 3: 14 pkt. Zadanie 4: 18 pkt.**
- *Punktacja: 46-50 pkt - bdb(5,0); 41-45 pkt - db+(4,5); 36-40 pkt - db(4,0); 31-35 pkt - dst+(3,5); 26-30 pkt - dst(3,0); 0-25 pkt - ndst (2,0).*
- Obowiązuje regulamin zajęć.
- Czas: 75 minut.
- **Egzamin należy wykonać na komputerach zamontowanych na stałe w pracowniach.**
- Student przesyłając rozwiązania oświadcza, że rozwiązał je samodzielnie.
- W trakcie egzaminu nie można korzystać z żadnych materiałów pomocniczych w żadnej formie poza tablicą znaków ASCII udostępnioną jako pdf na pendrive. Na pendrive znajduje się również folder do pierwszego zadania. Wszelkie kody powinny być napisane manualnie bez wspomagania się dodatkami automatycznie generującymi kod (np. Copilot, chat GPT itp.).
- Publikowanie poleceń i rozwiązań w internecie jest zabronione do czasu napisania egzaminu przez wszystkie osoby.
- Kod musi się kompilować bez błędów, aby był sprawdzany. Ostrzeżenia (tzw. warningi) są dopuszczalne, o ile nie prowadzi to do błędów merytorycznych. Użycie innego kompilatora niż gcc może powodować brak niektórych konstrukcji.
- Kod zakomentowany nie będzie sprawdzany.
- W trakcie egzaminu zostanie udostępniony przez prowadzącego pendrive. Zawartość pendrive będzie zawierać pliki pomocnicze do poleceń. Ten sam pendrive służy do zgrania rozwiązań. Umieszczenie rozwiązań na pendrive powinno odbyć się w czasie egzaminu.
- Rozwiązania po czasie mogą nie być sprawdzane.
- W rozwiązaniach należy przestrzegać dobrych praktyk i konwencji nazw stosowanej na wykładzie. W przypadku gdy zaburzenie zaleceń spowoduje niejednoznaczność wykonania kodu (tzw. unexpected behavior), za dane polecenie mogą być obniżone punkty (nawet do zera). Zalecane jest jawne dołączenie używanych bibliotek poprzez `#include`.
- Używanie typu `bool` w rozwiązaniach jest zakazane.
- Po kartkach z poleceniami można pisać i traktować jako brudnopis.
- Wszystkie zadania mają być rozwiązane w postaci aplikacji konsolowych "jednoplikowych" (bez podziału na pliki nagłówkowe).
- Format rozwiązania:
 - Zadania powinny być umieszczone w archiwum .zip na udostępnionym pendrive.
 - Nazwa archiwum powinna być wg schematu NUMERZESTAWU_NUMERALBUMU.zip gdzie numer zestawu znajduje się na górze kartki z poleceniami. np. A23_123456.zip
 - We wnętrzu archiwum powinny znajdować się tylko same kody w języku C, pliki powinny posiadać dokładnie nazwy (z uwzględnieniem wielkości znaków): `zad1.c`, `zad2.c`, `zad3.c`, `zad4.c`.
 - Sugerowana wersja języka C to C17.
 - Maksymalna waga archiwum 10 MB.
 - Należy nie dołączać plików wykonywalnego i plików tymczasowych kompilatora.
 - Archiwum powinno być bez hasła.
 - W przypadku pominięcia danego zadania, należy dodać plik o nazwie sprecyzowanej wyżej (zawartość może być pusta).

Polecenia są na odwrocie.

Zad.1. W folderze DebugXYZ (XYZ - losowe znaki) znajduje się projekt z kodem w języku C. W pliku main.c w niektórych liniach są komentarze. Twoim zadaniem jest wpisanie wartości odpowiednich zmiennych po wykonaniu konkretnej linii kodu. Dopisanie nowych linii czy zaburzenie struktury kodu oznacza zero punktów za polecenie. W przypadku znaków, należy zapisać sam znak w apostrofach np. 'c' (wielkość znaków ma znaczenie).

Zad.2. Napisz funkcję rekurencyjną, która dla otrzymanej w argumencie nieujemnej liczby całkowitej n zwraca wartość elementu o indeksie n ciągu zdefiniowanego w następujący sposób

$$\begin{aligned}a_0 &= a_1 = 1 \\ a_{2n} &= a_n, n > 0 \\ a_{2n+1} &= \lfloor \sqrt{2n} \rfloor, n > 0\end{aligned}$$

W zadaniu nie korzystaj z funkcji bibliotecznych poza instrukcjami wejścia/wyjścia. Stwórz przypadek testowy.

Wskazówka: $\lfloor x \rfloor$ oznacza część całkowitą z x czyli największą liczbę całkowitą nieprzekraczającą x .

Zad.3. Stwórz unię **Extra** przechowującą trzy pola: **x** typu **int**, **y** typu **double** oraz **z** typu **char**. Stwórz program zawierający tablicę 6 elementów typu **Extra**. Wypisz na konsoli zawartość tablicy.

Zad.4. Napisz funkcję, która przyjmuje jako argumenty dwie listy z głową o elementach typu:

```
struct elem {
    int t;
    struct elem * next;
};
```

o tej samej liczbie elementów. Funkcja ma zwrócić ile elementów na tych samych pozycjach są sobie równych. Stwórz przypadek testowy.

Przykład:

- Lista1: 3,4,5,6,7,8
- Lista2: 3,2,5,11,7,-7

Na tych samych pozycjach jest 3,5 i 7, więc funkcja powinna zwrócić 3.

