

Programowanie strukturalne (2024) - Egzamin (przykładowy) - Zestaw S05

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. Stwórz rekurencyjną funkcję, która przyjmuje jako argument napis. Funkcja ma zwrócić liczbę znaków będących dużymi literami w napisie. W zadaniu nie korzystaj z funkcji bibliotecznych poza instrukcjami wejścia/wyjścia. Stwórz przypadek testowy.

Uwaga: funkcja nierekurencyjna = 0pkt.

Zad.3. Zdefiniuj strukturę **Telefon** z polami **marka** (tablica znaków długości 30) oraz **iloscPołączeń** (typu **int**). Stwórz napisz funkcję **initTelefon**, która przyjmuje dwa argumenty: markę i ilość połączeń, i zwraca nowo utworzoną strukturę **Telefon** (jako wartość, nie wskaźnik) z polami ustawionymi na wartości przekazane jako argumenty. Funkcja powinna sprawdzać, czy marka ma długość co najmniej 3 i czy ilość połączeń jest większa niż 50. W przypadku nie spełnienia co najmniej jednego z tych warunków, funkcja powinna zwrócić strukturę z marką ustawioną jako "NIEZNANY" i ilością połączeń równą 100. Stwórz dwa przypadki testowe.

Zad.4. Napisz funkcję, która otrzymuje jako argument listę bez głowy o elementach typu:

```
struct node {
    int i;
    struct node * next;
};
```

Funkcja ma zdublować wartość ostatniego elementu, o ile lista jest nie pusta. W przypadku pustej listy, funkcja ma nic nie robić. Stwórz przypadek testowy.

Przykład: Dla listy 3,4,5 ma być zdublowana 5, więc po modyfikacji lista ma być postaci 3,4,5,5.

