Egzamin (pierwszy termin 2023) - Programowanie strukturalne - Zestaw E33

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).

Rozwiązania mają być umieszczone zgodnie ze specyfikacją:

- 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`.
- Maksymalna waga archiwum 10 MB.
- Archiwum powinno być bez hasła.
- W przypadku pominięcie danego zadania, należy dodać plik o nazwie sprecyzowanej wyżej (zawartość może być pusta).
- Kod zakomentowany nie będzie sprawdzany.

Za zachowanie specyfikacji dokładnie otrzymuję się dodatkowe 2 punkty. Zadania znacznie odbiegające od specyfikacji mogą nie być sprawdzane.

Polecenia są na odwrocie.

Zad.1. Dane są następujące wyrazy i znaki:

```
int int int suma tab a a [ ] ( ) , [ ]
```

Ułóż je we właściwej kolejności (zachowując podaną krotność), aby otrzymać nagłówek funkcji `suma`, która dostaje jako argumenty kolejno tablicę elementów i liczbę całkowitą. Następnie dodaj dowolną implementację funkcji i stwórz dla niej przypadek testowy.

Zad.2. Napisz funkcję, której argumentem są dwa napisy. Funkcja ma zwrócić 1 jeśli pierwszy napis jest wcześniej w porządku leksykograficznym niż drugi napis oraz 0 w pozostałych wypadkach. Stwórz przypadek testowy. Wykorzystaj typ `char`.

Wskazówka: Porządek leksykograficzny w napisach polega na porównywaniu poszczególnych znaków obu napisów od lewej do prawej, gdzie napis "mniejszy" jest ten, który ma znak o niższym kodzie ASCII na pierwszej niezgodnej pozycji.

Zad.3. Stwórz strukturę `Produkt` o dwóch polach: `nazwa` (napis) i `cena` (`float`). Następnie stwórz funkcję, której argumentami jest niepusta tablica struktur `Produkt` oraz rozmiar tablicy. Funkcja ma zwrócić "produkt" (jako strukturę) o najniższej cenie (w przypadku kilku takich, zwróć ostatni). Stwórz przypadek testowy.

Zad.4. Napisz funkcję, która otrzymuje jako argument listę z głową o elementach typu:

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

Funkcja ma wyświetlić na konsoli w kolejnych wierszach adresy elementów parzystych. Stwórz przypadek testowy.