Egzamin (pierwszy termin 2023) - Programowanie strukturalne - Zestaw E36

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:

```
float char int suma n x [ ] ( ) ,
```

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

Zad.2. Napisz funkcję `count_char()`, której argumentami są napis (typu `char*`) i znak (typu `char`). Funkcja powinna rekurencyjnie obliczyć i zwrócić liczbę wystąpień danego znaku w napisie. Stwórz przypadek testowy.

Uwaga. Funkcja nierekurencyjna = 0pkt.

Zad.3. Utwórz typ wyliczeniowy `Month` reprezentujący miesiące. Napisz funkcję `days_in_month()`, która przyjmuje jako argument wartość typu `Month` i zwraca liczbę dni w danym miesiącu. Pomiń problem roku przestępnego w tym zadaniu. Stwórz przypadek testowy dla funkcji.

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

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

i zwraca 1 jeśli suma kwadratów na obu listach jest sobie równa oraz zwraca 0 w pozostałych przypadkach. Stwórz jeden przypadek testowy.