Egzamin (pierwszy termin 2023) - Programowanie strukturalne - Zestaw E11

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:

```
char * char * int * int foo a b x ( , , )
```

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

Zad.2. Napisz funkcję o nazwie `findElement`, która przyjmuje jako argumenty tablicę liczb całkowitych, rozmiar tablicy, liczbę całkowitą `val` do znalezienia oraz wskaźnik na funkcję `isEqual`. Funkcja `isEqual` przyjmuje jako argumenty dwie liczby całkowite i zwraca wartość typu 0 lub 1, oznaczającą odpowiednio, czy liczby są równe. Funkcja `findElement` powinna przeszukiwać tablicę w celu znalezienia wartości `val` zgodnie z zasadami określonymi przez funkcję `isEqual`, a następnie zwrócić indeks tego elementu w tablicy (pierwszego napotkanego) lub `-1`, jeśli element nie istnieje. Stwórz przypadek testowy dla funkcji `findElement`.

Zad.3. Utwórz typ wyliczeniowy `DayOfWeek`, który reprezentuje dni tygodnia. Następnie napisz funkcję `printDayOfWeek`, która przyjmuje jako argument zmienną typu `DayOfWeek` i wypisuje na ekranie nazwę dnia tygodnia odpowiadającą danej wartości. Stwórz przypadek testowy dla funkcji.

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

```
struct node {
    float value;
    struct node * next;
};
```

zwraca adres ostatniej wartości dodatniej na liście. W przypadku pustej listy lub brak elementów dodatnich, funkcja ma zwrócić `NULL`. Stwórz jeden przypadek testowy.