Lab 6a (7 pkt) listy jednokierunkowe

Wszystkie etapy należy wykonać w podanej kolejności. Można tworzyć funkcje pomocnicze. Punktem startowym jest plik .c załączony do tego zadania. W treści zadania podane są nazwy punktów, a w załączonym pliku .c są wywołania tych funkcji. Należy samodzielnie dopasować nagłówki funkcji, by nie musieć zmieniać ich wywołań w main.

Programy operują na liście jednokierunkowej, której węzły przechowują pojedynczą liczbę typu int. Liczby mogą się powtarzać. Struktura węzła listy zdefiniowana została jako:

```
struct node
{
    int num;                    //dane
    struct node *nextptr;       //nastepny wezel
};
```

Num jest przechowywaną liczbą, nextptr wskazuje na kolejny element listy.

Etap 1 (3 pkt)

Przygotuj funkcje `nodeInsertatEnd`, `displayList`, `cleanList`.

Funkcja `nodeInsertatEnd` tworzy nowy węzeł i wstawia go na koniec listy. Nowy węzeł przechowuje wartość przekazaną jako drugi argument do tej funkcji. Głowa listy, do której wstawiany jest nowy element jest przekazana jako pierwszy argument.

Funkcja `displayList` wyświetla listę przekazaną jej jako argument zaczynając od głowy. Proszę ładnie sformatować wydruk konsoli, np.

```
Lista jednokierunkowa druga:
-----------------------------------------------
 2 --->   7 --->   6 --->   2 --->
-----------------------------------------------
```

Funkcja `cleanList` likwiduje całą listę przekazaną jej jako argument.

Etap 2 (2 pkt)

Przygotuj funkcję `sortRange`.

Funkcja `sortRange` sortuje listę przekazaną jej jako argument w zakresie od elementu o indeksie zero (czyli głowa) do elementu o indeksie podanym jako drugi argument włącznie. Program będzie testowany dla różnych poprawnych wartości zakresu, czyli od `0` do `FL-1`.
UWAGA: nie wolno alokować dodatkowej pamięci ani tworzyć pomocniczej listy. Funkcja sortuje listę manipulując węzłami, a nie wartościami w węzłach.

Etap 3 (2 pkt)

Przygotuj funkcję `sweepList`.

Funkcja `sweepList` usuwa z listy i dealokuje pamięć duplikatów, czyli ponownych wystąpień danej liczby w liście przekazanej jej jako argument.
UWAGA: nie wolno tworzyć pomocniczej listy przy usuwaniu duplikatów ani sortować listy (kolejność wystąpienia unikalnych wartości musi być zachowana)!

Przykładowy wydruk z programu po stworzeniu listy:

```
Singly linked list:
-------------------------------------------
5 --->   0 --->   2 --->   9 --->   4 --->   5 --->   8 --->   9 --->   9
--->   3 --->   2 --->   7 --->   2 --->   9 --->   1 --->   0 --->   0 --
->   7 --->
```

Przykładowy wydruk z programu po posprzątaniu listy:

```
No data found in the empty list.
```

Przykładowy wydruk z programu przed i po posortowaniu elementów z danego zakresu:

```
Singly linked list (new one):
-------------------------------------------
2 --->   4 --->   0 --->   4 --->   4 --->   2 --->   3 --->   0 --->   2
--->   2 --->   1 --->   0 --->   3 --->   2 --->   2 --->   2 --->   2 --
->   4 --->

Singly linked list after sorting in the range 0 - 5:
0 --->   2 --->   2 --->   4 --->   4 --->   4 --->   3 --->   0 --->   2
--->   2 --->   1 --->   0 --->   3 --->   2 --->   2 --->   2 --->   2 --
->   4 --->

Singly linked list without duplicates:
-------------------------------------------
0 --->   2 --->   4 --->   3 --->   1 --->
```