

Lista 3 – wskaźniki (17 marca)

W ramach następującej definicji elementu listy pojedynczo związanej:

```
struct Link
{
    int value;
    Link* next = nullptr;
};
```

1. Zaimplementuj i przetestuj funkcje:

- a. `size_t size(const Link* head);`
- b. `size_t sum(const Link* head);`

Pierwsza z nich zwraca liczbę elementów listy, druga - sumę wartości przechowywanych na liście

2. Funkcja **release** zwalnia pamięć, jaką zajmują elementy listy wskazywanej przez **phead**:

```
void release(Link** phead)
{
    Link* head = *phead;
    while (head != nullptr)
    {
        Link* tmp = head;
        head = head->next;
        delete tmp;
    }
    *phead = nullptr;
}
```

Zaimplementuj tę samą funkcję przy pomocy rekurencji (bez pętli).

3. W pewnej bardzo znanej firmie w połowie lat 90. podczas rozmowy kwalifikacyjnej proszono kandydatów o zaimplementowanie funkcji **reverse**, która odwraca kolejność elementów na liście. Czyli np. listę `1->2->5->9->nullptr` zamienia na listę `9->5->2->1->nullptr`. Zaimplementuj tę funkcję.

Wskazówka. To *jest* trudne zadanie (jak na ten etap nauki). Możesz poszukać informacji o sposobach jego rozwiązywania w internecie lub podręcznikach. Niemniej, uniesiesz się honorem i nie przepisziesz ich z tamtych źródeł. Poprawne rozwiązanie nie może „dotknąć” danych - one muszą pozostać tam, gdzie były. Całe rozwiązanie opiera się na żonglerce wskaźnikami. Pamiętaj, że w problemach tego rodzaju Twoim najlepszym przyjacielem jest kartka i ołówek, a tuż za nimi stoi debugger.