Laboratorium 2B (7pkt)

Przygotowany jest szkielet programu wraz z funkcją **main**. W **main** zadeklarowane jest przykładowe użycie funkcji, które mają zostać zaimplementowane w kolejnych etapach.

Załóżmy, że N = 100 i jest to rozmiar statycznej tablicy jednowymiarowej, na której wykonywać będziemy operacje. Operacje będą wykonywane tylko na części tej tablicy. Zakres, do którego operujemy określać będziemy zmienną o nazwie „n". W funkcji **main** zadeklarowane są 3 tablice C1, C2, C3 oraz odpowiadające im zakresy n1, n2, n3.

Etap 1 (2 pkt)
Należy zaimplementować 3 funkcje, które można wywołać następująco:

- fill_random(C, n, range)
Funkcja fill_random wypełnia pierwsze n elementów w 1-wymiarowej tablicy T (n<=N) losowymi wartościami w zakresie od 0 do range-1. Na przykład dla wywołania: fill_random(C2, 7, 10) tablica C2 jest wypełniona następującymi wartościami: [9 8 7 0 7 7 9].

-fill_partially(C, n, value1, n1, value2)
Funkcja fill_partially wypełnia pierwsze n elementów w 1-wymiarowej tablicę T (n<=N) podaną stałą value1 na pozycjach od 0 do n1-1 oraz stałą value2 dla elementów w tablicy na indeksach od n1 do n-1. Na początku funkcji sprawdź czy parametry są poprawne. Na przykład dla wywołania: fill_partially(C1, 10, 0, 1, 1): C1: [0 1 1 1 1 1 1 1 1 1]

- print(C, n)
Funkcja print wypisuje na ekran elementy 1-wymiarowej tablicy W o rozmiarze n (n<=N) w formacie [3 5 7 9]

Etap 2 (1 pkt)
Należy zaimplementować funkcję, o następującym nagłówku:
int sum(int C[N], int n, int from, int to)
Funkcja sum dodaje odpowiednie elementy 1-wymiarowej tablicy C o rozmiarze n (n<=N), zaczynając od elementu na indeksie from a kończy na indeksie to – 1 (Nie dodajemy końca przedziału). Obliczona suma jest zwracana. Przykładowo dla tablicy: C = [1,2,3,4,5]
sum(T, 5, 1, 3) zwróci liczbę 2 + 3 = 5.

Etap 3 (1 pkt)
Należy zaimplementować funkcję, którą można wywołać następująco: Fibonacci(T, n);
Funkcja fibonacciego iteracyjnie dodaje odpowiednie elementy 1-wymiarowej tablicy T o rozmiarze n (n<=N), zaczynając od elementu na indeksie 2, a kończąc na indeksie n-1. Tworzymy go w następujący sposób:
- Wartość na indeksie 0 musi mieć wartość 0
- Wartość na indeksie 1 musi mieć wartość 1
- Każda kolejna wartość w tablicy tworzymy na podstawie wzoru: T[i] = T[i-1] + T[i-2]
Przykładowo tablica o rozmiarze 6 będzie wyglądać następująco [0,1,1,2,3,5]

Etap 4 (1 pkt)
Należy zaimplementować funkcję, którą można wywołać następująco: check(T, n);
Funkcja check sprawdza, czy elementy 1-wymiarowej tablicy T o rozmiarze n (n<=N) tworzą ciąg Fibonacciego. Zwraca ostatni element tego ciągu, gdy wykryto ciąg Fibonacciego, wpp zwraca -1

Laboratorium 2B (7pkt)

Etap 5 (1 pkt)

Należy zaimplementować funkcję, którą można wywołać następująco: fibonacci_depth(T, n, d); Funkcja fibonacci_deepth liczy ciąg Fibonacciego rzędu d, to znaczy że funkcja iteracyjnie dodaje odpowiednie elementy 1-wymiarowej tablicy T o rozmiarze n (n<=N), zaczynając od elementu na indeksie d a kończy na indeksie n-1.
Ciąg Fibonacciego rzędu d tworzymy w następujący sposób:
- Wartość na indeksach od 0 do d - 2 muszą mieć wartość 0
- Wartość na indeksie d - 1 musi mieć wartość 1
- Każda kolejna wartość w tablicy tworzymy na podstawie wzoru: T[i] = T[i-1] + T[i-2] + ... T[i-d]

Przykładowo tablica o rozmiarze 7 dla sekwencji rzędu 3 będzie wyglądać następująco:
[0,0,1,1,2,4, 7]

Etap 6 (1 pkt)

Należy zaimplementować funkcję, którą można wywołać następująco:
merge(T1, n1_from, n1_to, T2, n2_from, n2_to, T3, n3); Funkcja merge łączy odpowiednie elementy 1-wymiarowych tablic T1 oraz T2. Elementy z tablicy T1 zaczynając od n1_from do n1_to są zapisane do tablicy T3 następnie elementy z tablicy T2 od n2_form do n2_to są dopisane do tablicy T3. Wszystkie pozostałe wartości w T3 są wypełnione jako zera. Sprawdź poprawność danych wejściowych. Przykład:
T1 = [0,1,2,3,5,8], n1_from=2, n1_to = 4
T2 =[0,0,1,1,2,4, 7] n2_from=0, n2_to = 3
Wynik: T3=[2,3,5,0,0,1,1]