

**Zadanie 1** (0.5 pkt) Zaimplementuj algorytm sortowania przez wstawianie dla ciągu długości  $n$ .

```
insertionsort(A)
  for i = 1 to length(A) - 1
    x = A[i]
    j = i - 1
    while j >= 0 and A[j] > x
      A[j+1] = A[j]
      j = j - 1
    A[j+1] = x
  End
```

**Zadanie 2** (0.5 pkt) Zaimplementuj algorytm sortowania przez scalanie dla ciągu  $A$  długości  $n$ . Funkcja `merge` służy do łączenia dwóch posortowanych ciągów w jeden posortowany ciąg liczb.

```
mergesort(A, a, b)
  if a < b:
    c = (a+b)/2
    mergesort(A, a, c)
    mergesort(A, c+1, b)
    merge(T, a, c, b)
  End
```

**Zadanie 3** (1 pkt) Porównaj czasową złożoność obliczeniową algorytmów: sortowania przez wstawianie i sortowania przez scalanie. W celu uzyskania miarodajnych wyników wykonaj wiele ( $> 10^2$ ) iteracji. W każdej iteracji wylosuj ciąg o stałej długości ( $> 10^5$ ). Zmierz dla obu algorytmów czas wykonania wszystkich iteracji, czas najszybszej i najwolniejszej iteracji, policz średni czas wykonania iteracji.

Podczas implementacji sortowania przez scalanie należy zwrócić uwagę na ograniczenia wielkości stosu co przekłada się na ilość możliwych poziomów rekurencji.

---