

# Lista zadań nr 1

## Zadanie 1.

Przeanalizuj poniższą sekwencję wyrażeń. Jaki wynik wypisze interpreter w odpowiedzi na każde z nich, zakładając, że będą obliczane w kolejności w której są podane? Sprawdź swoje przewidywania używając interpretera.

```
10
```

```
(+ 5 3 4)
```

```
(- 9 1)
```

```
(/ 6 2)
```

```
(+ (* 2 4) (- 4 6))
```

```
(define a 3)
```

```
(define b (+ a 1))
```

```
(+ a b (* a b))
```

```
(= a b)
```

```
(if (and (> b a) (< b (* a b)))  
    b  
    a)
```

```
(cond [(= a 4) 6]  
      [(= b 4) (+ 6 7 a)]  
      [else 25])
```

```
(+ 2 (if (> b a) b a))
```

```
(* (cond [(> a b) a]
```

```

      [( < a b) b]
      [else      -1])
(+ a 1))

```

**Zadanie 2.**

Przedstaw w postaci prefiksowej poniższe wyrażenie:

$$\frac{5 + 4 + (2 - (3 - (6 + \frac{4}{5})))}{3(6 - 2)(2 - 7)}.$$

**Zadanie 3. (2 pkt)**

Zastosuj zasady obliczania wyrażeń poznane na wykładzie do obliczenia wartości poniższych wyrażeń. Które z nich spowodują błąd i dlaczego?

```
(* (+ 2 2) 5)
```

```
(* (+ 2 2) (5))
```

```
(*(+ (2 2) 5))
```

```
(*(+ 2
      2) 5)
```

```
(5 * 4)
```

```
(5 * (2 + 2))
```

```
((+ 2 3))
```

```
+
```

```
(define + (* 2 3))
```

```
+
```

```
(* 2 +)
```

```
(define (five) 5)
```

```
(define four 4)
```

```
(five)
```

```
four
```

```
five
```

```
(four)
```

#### Zadanie 4. (2 pkt)

Zdefiniuj procedurę o trzech argumentach będących liczbami, której wynikiem jest suma kwadratów dwóch większych jej argumentów. Napisz dla tej procedury testy jednostkowe z użyciem biblioteki `rackunit`.

#### Zadanie 5.

Zauważ że w naszym modelu obliczania wartości dopuszczamy, aby operatorami były wyrażenia złożone. Korzystając z tej obserwacji, wyjaśnij działanie następującej procedury:

```
(define (a-plus-abs-b a b)
  ((if (> b 0) + -) a b))
```

#### Zadanie 6. (2 pkt)

Z wykładu wiemy, że `and` i `or` są formami specjalnymi, których reguły obliczania są następujące:

- Forma `and` oblicza podwyrażenia od lewej do prawej tak długo, aż natrafi na wartość `#f`, którą wtedy zwraca. Jeśli żadne podwyrażenie nie oblicza się do `#f`, zwracana jest wartość ostatniego podwyrażenia.
- Forma `or` oblicza podwyrażenia od lewej do prawej tak długo, aż natrafi na wartość inną niż `#f`, którą wtedy zwraca. Jeśli wszystkie podwyrażenia obliczają się do `#f`, zwracana jest wartość ostatniego podwyrażenia.

Przykładowo, wyrażenia `(and #t 1)`, `(or #f 1)` oraz `(or 1 2)` są obliczane do wartości 1.

Wykorzystaj to, aby za pomocą form specjalnych **and** i **or** skonstruować wyrażenie równoważne pod względem zachowania formie specjalnej **if**. Dokładniej, dla wyrażenia postaci (**if** ifCond ifTrue ifFalse), należy skonstruować równoważne pod względem zachowania wyrażenie wykorzystujące formy specjalne **and** i **or** oraz wyrażenia ifCond, ifTrue oraz ifFalse. Można założyć, że wyrażenia ifTrue oraz ifFalse nigdy nie obliczają się do wartości boolowskich.

### Zadanie 7. (3 pkt)

Na podstawie kodu przykładowego z wykładu rysującego animację lądującej rakiety, zaimplementuj własną animację, która przedstawia odliczanie i start rakiety. Aby przedstawić odliczanie, zastosuj procedurę `text`, której argumentami są: ciąg znaków do narysowania (możesz użyć procedury `number->string`), liczba oznaczająca rozmiar czcionki, oraz ciąg znaków oznaczający kolor (np. `"black"`). Ruch rakiety powinien odbywać się ze stałym przyspieszeniem. Można posłużyć się znanym z lekcji fizyki wzorem  $y = \frac{1}{2}at^2$ . Zadbaj o czytelność swojej implementacji!