Home task:

- Install and play with Racket (drracket — Racket IDE; racket language documentation);
- Implement FlowChart interpreter on Racket, $int^{FlowChart}_{Racket}$, with respect to the abstract syntax from Fig. 1; FlowChart program example can be found on Fig. 3a;
- Implement Post's variant Turing Machine interpreter on FlowChart, $int^{TM}_{FlowChart}$, with respect to the abstract syntax from Fig. 2; TM program example can be found on Fig. 3b.

$\mathscr{X}$	::=	$\{x, y, z, \dots\}$	Variable
$\mathscr{P}$	::=	**read** $\mathscr{X}^*$; $\mathscr{B}^+$	Program
$\mathscr{B}$	::=	$\mathscr{L}$: $\mathscr{A}^*$ $\mathscr{J}$	Basic block
$\mathscr{A}$	::=	$\mathscr{X}$:= $\mathscr{E}$	Assignment
$\mathscr{J}$	::=	**goto** $\mathscr{L}$	Jump
	|	**if** $\mathscr{E}$ **goto** $\mathscr{L}$ **goto** $\mathscr{L}$	
	|	**return** $\mathscr{E}$	
$\mathscr{E}$	::=	$\mathscr{C}$ | $\mathscr{X}$ | $\mathscr{O}$ $\mathscr{E}^*$	Expression
$\mathscr{C}$	::=	'*Value* /* a quoted value */	Constant
$\mathscr{O}$	::=	car | cdr | cons | ... /* (any other) */	Operator
$\mathscr{L}$	::=	/* whatever */	Label

Fig. 1: Abstract FlowChart syntax

$\mathscr{A}$	::=	$\{0, 1, \sqcup\}$	Alphabet
$\mathscr{P}$	::=	$\mathscr{I}^*$	Program
$\mathscr{I}$	::=	**left** | **right** | **write** $\mathscr{A}$ | **goto** i | **if** $\mathscr{A}$ **goto** i	Instruction where i is instruction index

Fig. 2: Abstract Turing Machine syntax

```
read name namelist valuelist;
search: if (equal? name (car namelist)) goto found goto cont;
cont  : valuelist := cdr valuelist;
        namelist  := cdr namelist ;
        goto search;
found : return car valuelist;
```

(a) FlowChart program example (find name)

```
0: if 0 goto 3
1: right
2: goto 0
3: write 1
```

(b) Turing Machine program example

Fig. 3: Program examples (in some concrete syntax)

1 How exactly should it look like on Racket

First, you have to define a FlowChart interpreter on Racket. I.e. something like:

```
(define int ((lambda program data) (<interpreter code>)))
```

where *program* stands for FlowChart program and *data* — for a list of its input data. Note that both *program* and *data* are data, i.e. quoted values, from the Racket's point of view. In order to avoid parsing we write programs on FlowChart as their abstract syntax trees. E.g. program *find_name* from Fig. 3a can be defined as follows

```
(define find_name
  '((read name namelist valuelist)
   (search (if (equal? name (car namelist)) found cont))
   (cont (:= valuelist (cdr valuelist))
         (:= namelist (cdr namelist))
         (goto search))
   (found (return (car valuelist)))
   ))
```

and its invokation

```
(int find_name '(y (x y z) (1 2 3))
;; the expected output is '2
```

Then, you have to define a Turing-machine interpreter on FlowChart

```
(define tm-int
  '((read Q Right)
    (init (:= Qtail Q) (:= Left '()) (goto loop))
    (loop (if (...) label-1 label-2))
    ...
   )
  )
```

An example of TM-program definition:

```
(define tm-example '((0 if 0 goto 3) (1 right) (2 goto 0) (3 write 1)))
```

and its invocation

```
(int tm-int `(,tm-example (1 1 1 0 1 0 1)))
;; the expected output --- '(1 1 0 1)
```