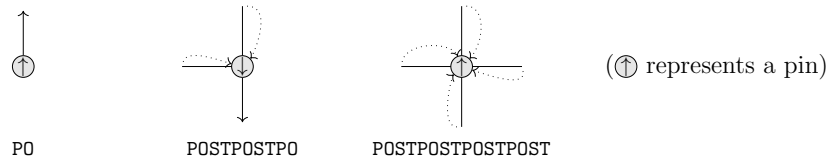
2 Compiler Construction (jdy22)

PINPEN programs control a pen and a set of pins on a plane. The pin and pens each have a position and direction. Programs are sequences of the following commands:

`P`	Places a new pin with the pen's current position and direction.
`O`	Moves the pen one step. Also draws a line from the old position to the new position.
`S`	Sets the pen's position and direction to the last-placed pin. Removes the pin. Does not draw anything.
`T`	Turns the pen 90° anti-clockwise.

For example, the program `POSTPOSTPOSTPOST` draws a `+` symbol as shown below:



(*a*) Given the following API:

```
type pos = int * int            val step : dir -> pos -> pos
type dir                        val draw : pos -> pos -> unit
type command = P | O | S | T    val turn : dir -> dir
```

write a PINPEN interpreter with type `command list -> pos * dir -> unit` using exceptions to implement `P` and `S`, and without using references. The second argument represents the pen's position and direction. [6 marks]

(*b*) Convert your PINPEN interpreter to use a continuation argument rather than exceptions to implement `P` and `S`, giving the interpreter the following type:

```
command list -> pos * dir -> (command list -> unit) -> unit
```

[6 marks]

(*c*) State whether each of the following PINPEN optimisations is valid:

(i) `PPOSS` $\rightsquigarrow$ `POS` (ii) `POS` $\rightsquigarrow$ `O` [2 marks]

(*d*) Describe *all* the sequences of commands that can be removed from *any* PINPEN program without changing the program's behaviour. [6 marks]