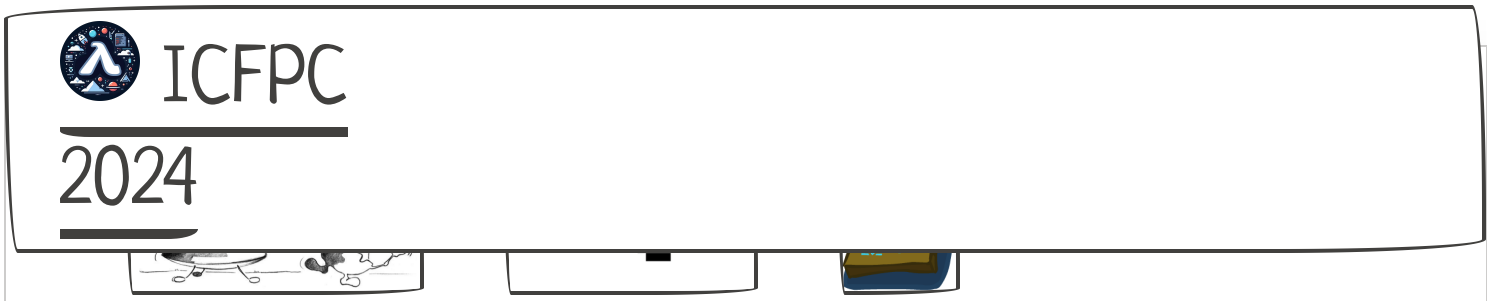


ICFP language

An *Interstellar Communication Functional Program* (ICFP) consists of a list of space-separated *tokens*. A *token* consists of one or more printable ASCII characters, from ASCII code 33 (`!`) up to and including code 126 (`~`). In other words, there are 94 possible characters, and a *token* is a nonempty sequence of such characters.

The first character of a *token* is called the *indicator*, and determines the type of the *token*. The (possibly empty) remainder of the *token* is called *body*. The different *token* types are explained in the next subsections.

Booleans

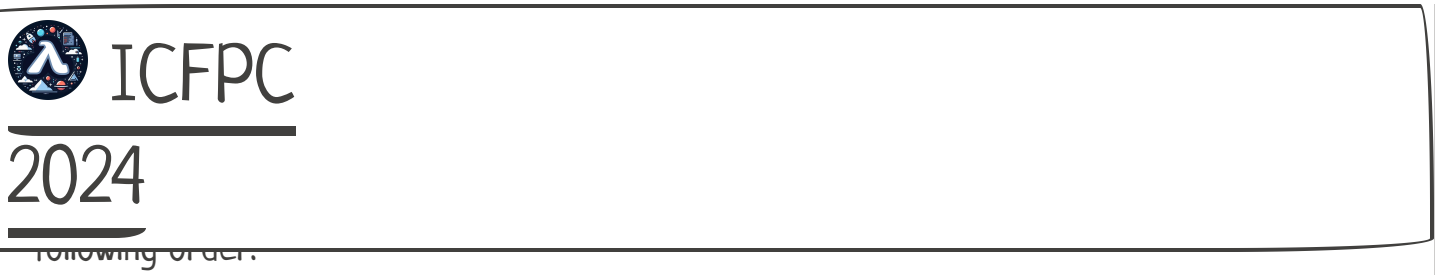
`indicator = T` and an empty *body* represents the constant `true`, and `indicator = F` and an empty *body* represents the constant `false`.

Integers

`indicator = I`, requires a non-empty *body*.

The *body* is interpreted as a base-94 number, e.g. the digits are the 94 printable ASCII characters with the exclamation mark representing `0`, double quotes `1`, etc. For example, `I/6` represent the number `1337`.

Strings



following order:

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!"#$%
```

Here `<space>` denotes a single space character, and `<newline>` a single newline character. For example, `SB%,,/}Q/2,$_` represents the string "Hello World!".

Unary operators

`indicator = U`, requires a *body* of exactly 1 character long, and should be followed by an ICFP which can be parsed from the tokens following it.

Character	Meaning	Example
`-`	Integer negation	`U- I$ -> -3`
`!`	Boolean not	`U! T -> false`
`#`	string-to-int: interpret a string as a base-94 number	`U# S4%34 -> 15818151`
`$`	int-to-string: inverse of the above	`U$ I4%34 -> test`

The `->` symbol in this table should be read as "will evaluate to", see Evaluation.

Binary operators

`indicator = B`, requires a *body* of exactly 1 character long, and should be followed by two ICFPs (let's call them `x` and `y`).

	~~Integer subtraction~~	~~B- I$ I# -> 1~~
`*`	Integer multiplication	`B* I$ I# -> 6`
`/`	Integer division (truncated towards zero)	`B/ U- I( I# -> -3`
`%`	Integer modulo	`B% U- I( I# -> -1`
`<`	Integer comparison	`B< I$ I# -> false`
`>`	Integer comparison	`B> I$ I# -> true`
`=`	Equality comparison, works for int, bool and string	`B= I$ I# -> false`
`|`	Boolean or	`B| T F -> true`
`&`	Boolean and	`B& T F -> false`
`.`	String concatenation	`B. S4% S34 -> "test"`
`T`	Take first `x` chars of string `y`	`BT I$ S4%34 -> "tes"`
`D`	Drop first `x` chars of string `y`	`BD I$ S4%34 -> "t"`
`$`	Apply term `x` to `y` (see Lambda abstractions)	

If

ICFPC 2024

evaluates to `no`.

Lambda abstractions

`indicator = L` is a lambda abstraction, where the *body* should be interpreted as a base-94 number in the same way as integers, which is the variable number. `indicator = v` is a variable, with again a *body* being the base-94 variable number.

When a lambda abstraction appears as the first argument of the binary application operator `$`, the second argument of the application is assigned to that variable. For example, the ICFP

```
B$ B$ L# L$ v# B. SB%,,/ S}Q/2,$_ IK
```

represents the program (e.g. in Haskell-style)

```
((\v2 -> \v3 -> v2) ("Hello" . " World!")) 42
```

which would evaluate to the string `"Hello World!"`.

Evaluation

The most prevalent ICFP messaging software, Macroware Insight, evaluates ICFP messages using a call-by-name strategy. This means that the binary application operator is non-strict; the second argument is substituted in the place of the binding variable (using capture-avoiding substitution). If an argument is not used in the body of the lambda abstraction, such as `v3` in the above example, it is never evaluated. When a variable is used several times, the expression is evaluated multiple times.

ICFPC 2024

```
B+ I' B* I$ I#
B+ I' I'
I-
```

Limits

As communication with Earth is complicated, the Cult seems to have put some restrictions on their Macroware Insight software. Specifically, message processing is aborted when exceeding `10_000_000` beta reductions. Built-in operators are strict (except for `B$`, of course) and do not count towards the limit of beta reductions. Contestants' messages therefore must stay within these limits.

For example, the following term, which evaluates to `16`, uses `109` beta reductions during evaluation:

```
B$ B$ L" B$ L# B$ v" B$ v# v# L# B$ v" B$ v# v# L" L# ? B= v# I! I"
```

Researchers expect that the limit on the amount beta reductions is the only limit that contestants may run into, but there seem to also be some (unknown) limits on memory usage and total runtime.

Unknown operators

The above set of language constructs are all that researchers have discovered, and it is conjectured that the Cult will never use anything else in their communication towards Earth. However, it is unknown whether more language constructs exist.