BOOP: How to Right Code!

A novel problem-solving tool for students of introductory computer science. This is designed to promote structured thinking, strong design choices, self-reflection and a well-probed problem-solving stream.

Language Syntax

The ICS language syntax is designed to be simple and intuitive, allowing users to write structured documents easily. There are only 5 syntax features to know:

> - **Sections** follow tag-based hierarchy (similar to HTML!).
> - **Main-tags** (intuitively, these are the 'broad' sections) use double angle brackets (`<< >>`).
> - **Sub-tags** (intuitively, these are the sub-sections) use a simple `tag_name:` format
> - A main-tag MUST have all its sub-tags, even if they are empty. This is to ensure that the document is well-structured and complete.
> - **Comments** are started using `//`.

Header Block

Every submission must inlcude a header in the following format. All fields of the header are mandatory.

```
<<header

assignment: Assignment 5: Sorting & Searching
student: Vaani Goenka
date: 15th June 2025
collaborators: None

header>>
```

Syntax in action: BOOP!

---

1. Problem Block

```
<<problem 1: "Title of the Problem"

... (all other BOOP blocks go here)

problem>>
```

> - Every problem must be enclosed within a problem block. Failure to do so might result in a blank submission despite other BOOP tags.
> - 'problem' must be followed by a number, a colon and the title of the problem in double quotes as shown above.
> - Proper error message for the above is in progress.

---

2. Blueprint

This main-section is intended for the user to lay out the correctness criteria of their program i.e. what are the characeteristics (formally) of the input and output that must hold true for the program to be considered correct.

```
<<blueprint
requires: ...
ensures:...

blueprint>>
```

---

3. Operational Steps

This section is for the user to explain their algorithm (process) for solving the given problem as if to a human.

```
<<operational steps
step 1: Considering the base case n=0, we can think of the problem as one....
step n:...

operational steps>>
```

---

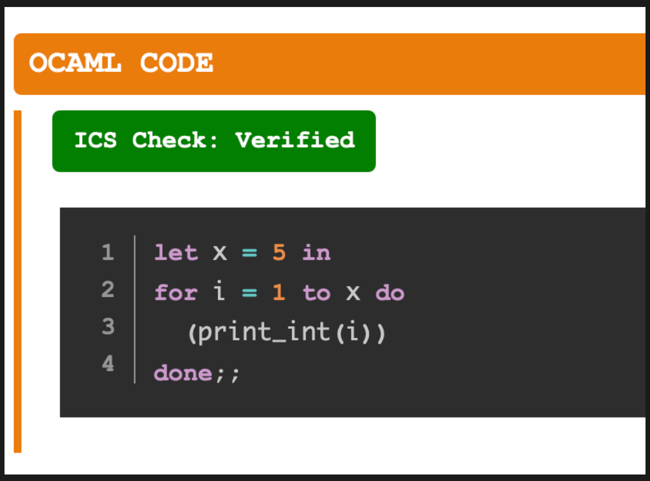
4. Ocaml Code

Write your standard ICS-friendly OCaml code here. It will be validated by the ICS OCaml validator.

```
<<ocaml code
(* Write your code in regular OCaml syntax, following ICS guidelines *)

ocaml code>>
```

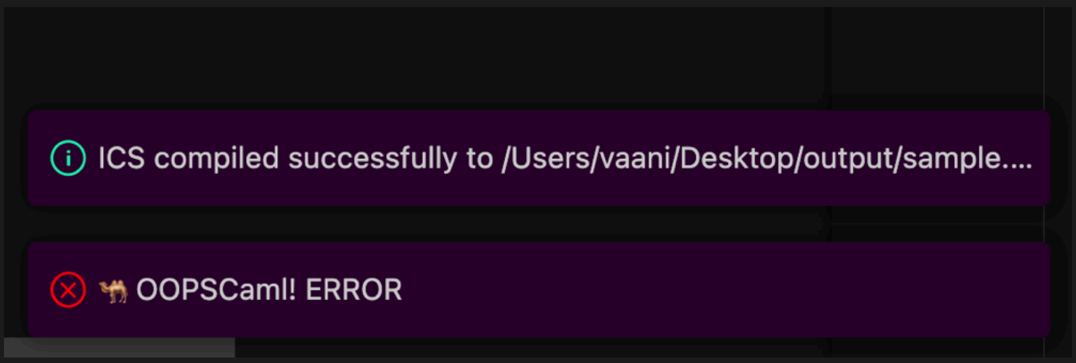
The status of your code is then reflected in the compiled version as so:


OCAML CODE

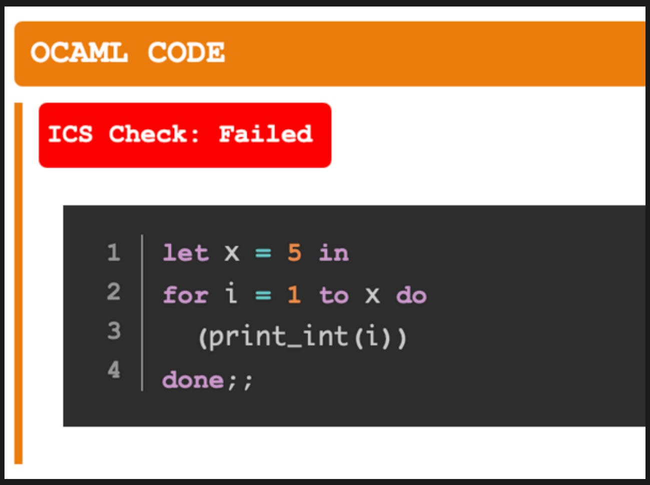
ICS Check: Verified

```ocaml
let x = 5 in
for i = 1 to x do
  (print_int(i))
done;;
```

Errors in your code will appear in a pop-up during compilation, but will not stop compilation. Your submission will be compiled and your PDF will have your code marked as invalid according to ICS syntax.


ⓘ ICS compiled successfully to /Users/vaani/Desktop/output/sample....

ⓧ 🐫 OOPSCaml! ERROR


OCAML CODE

ICS Check: Failed

```ocaml
let x = 5 in
for i = 1 to x do
  (print_int(i))
done;;
```

---

5. 📖 Proof

This section is intended for the user to provide a complete and formal proof that their program satisfies the given correctness criteria in the blueprint.

Proof by induction would look like:

```
<<proof
<<induction

base case: ...

induction hypothesis: ...

indutive step: ...

induction>>

proof>>
```

Loop invariants would look like:

```
<<proof
<<invariant

pre-condition: ...

after the ith step: ...

after the (i+1)th step: ...

post-condition: ...

invariant>>

proof>>
```

Usage:

1. Install dependencies in your working folder (Assuming you have opam installed. opam is the package manager for OCaml.)

```
opam install dune ocaml odoc ppxlib
```

2. Create your .ics file in your working folder
3. Open the command palette (Cmd+Shift+P on mac or Ctrl+Shift+P on windows)
4. Type ICS
5. Select ICS: Compile ICS
6. You can view your pdf open in your default browser. Further, a raw html file (with some accompanying css) will be available on `Desktop/output`. Most users may discard this file if you do not need it.

Please note that the dropdown option: 'Compile for assignment #1' is a test option and not relevant to current users!

Acknowledgements

Thank you to Prof Aalok D. Thakkar for his guidance and mentorship.

Version Info

Developer and Maintainer: Vaani Goenka vaani.goenka_ug2024@ashoka.edu.in
Changelog
0.5.0
Second major release
OCaml code validation
Instructor discretion to add plugins to customise validation
Better extension features like collapsing sections
Better error messages