A simple J0 Program

```
module M {              // module
    int i;              // global variable

    void foo() {        // function
        int k;          // local variable
        k = 0;
        i = k;
    }

    int inc(int x) {
        return x + i;
    }

    int j;
}
```


J0 Namespaces

A defined name is visible,

- if the declaration is in the same or an enclosing block, and
- if the name is declared textually before.

If there is more than one declaration in the enclosing blocks, then only the innermost declaration is visible.

```
module M {
    int i;
    int foo(int n) {
        if (n > 0) return foo(n - 1); else return i;
    }
    void init() {
        int i;
        i = foo(1);
    }
}
```


J0 Types

J0 offers two built-in primitive types:

- `boolean`, and
- `int`.

Furthermore J0 supports one-dimensional arrays of any type; e.g.

- `int[]`
- `boolean[][]`

J0 Arrays

- Arrays are created with a special form of a variable declaration. Here is a definition of an array of type `int[]`:

  ```
  int js[3];   // variable declaration and array creation of
               // a one-dimensional integer array of length 3
  ```

- It is possible to define multi-dimensional arrays (arrays of arrays):

  ```
  int ms[3][4];   // defines a 3 x 4 matrix of type int[ ][ ]
  ```

- The array indices of an array of length n range from 0 to $n-1$
- An array element can be accessed with the `[...]` operator:

  ```
  int as[3];
  ...
  as[2] = as[0] + as[1];
  ```

- Restriction: it is not possible to assign values of an array type

  ```
  int bs[3];
  bs = as;
  ```

J0 Functions

- Parameter passing is done
  - call-by-value for values of type `int` and `boolean`, and
  - call-by-reference for arrays.

```
void setOne(int[ ] xs) { xs[0] = 1; }
void setTwo(int x) { x = 2; }
int run() {
    int ys[1];
    ys[0] = 0;
    setOne(ys);
    setTwo(ys[0]);
    return ys[0];
}
```

- Functions cannot return arrays. The result type of a function can either be `void`, `int` or `boolean`.
- Functions with result type `void` do not return any result.

Short-Cut Evaluation

- Generally, before executing a binary operation, both operands are evaluated first.
- For the logical operators `&` and the `|`, short-cut evaluation is performed; i.e.
  - In expression e_1 & e_2, `e2` is only evaluated, if `e1` evaluates to `true`,
  - in expression e1 | e2, `e2` is only evaluated, if `e1` evaluates to `false`

```
int i;
void bar() { return i < 5; }
int foo(boolean b) {
    i = 0;
    if (b & bar()) return i; else return i + 2;
}
```

What does the function call foo(false) return?