El lenguaje escogido por mí es Haxe. Es un lenguaje de alto nivel, fuertemente tipado que puede ser compilado a multiples programas. Como JavaScript, Python, Php, entre otros. Es orientado a objetos también, permitiendo la creación de clases, interfaces, clases abstractas, herencias (Solo se puede heredar de una clase). Posee tipos de datos, float, int y bool. Dentro de las clases, solo se permite un constructor, los cual puede ser inconveniente en muchas ocasiones, también permite en cierto tipo de estructura llamadas abstract hacerle overload a los operadores. La tabla de precedencia es la siguiente:

Precedence

In order of descending precedence (i.e. operators higher in the table are evaluated first):

Operators	Note	Associativity
`!`, `++`, `--`	postfix unary operators	right
`~`, `!`, `-`, `++`, `--`	prefix unary operators	right
`%`	modulo	left
`*`, `/`	multiplication, division	left
`+`, `-`	addition, subtraction	left
`<<`, `>>`, `>>>`	bitwise shifts	left
`&`, `|`, `^`	bitwise operators	left
`==`, `!=`, `<`, `<=`, `>`, `>=`	comparison	left
`...`	interval	left
`&&`	logical and	left
`||`	logical or	left
`@`	metadata	right
`?:`	ternary	right
`%=`, `*=`, `/=`, `+=`, `-=`, `<<=`, `>>=`, `>>>=`, `&=`, `|=`, `^=`	compound assignment	right
`=>`	arrow	right

El lenguaje posee asociación dinámica de métodos, y ofrece algo parecido a una asociación dinámica con la posibilidad de colocarle a las variables y métodos la etiqueta final que permite que una vez asignado un valor a dicha variable, o definido el método no se pueda cambiar más.

since Haxe 4.0.0

The `final` keyword can be used on class fields with the following effects:

- `final function ...` to make a function non-overridable in subclasses.
- `final x = ...` to declare a constant that must be initialized immediately or in the constructor and cannot be written to.
- `inline final x = ...` is the same but inlines the value wherever it is used. Only constant values can be assigned.

Por más de ser un lenguaje de uso no tan a gran escala ofrece una muy buena cantidad de documentación. Esta contiene ejemplos detallados, así como explicaciones de casi todas las funciones y o propiedades del lenguaje. https://haxe.org/manual/introduction.html

Desde hace poco Haxe implementó como sistema de concurrencia los Threads y Process, aunque al ser tan reciente la documentación provista no es tan a gran escala. Aun así me fue posible desarrollar los algoritmos. Para declarar los Threads es necesario llamarlos como aparece en la imagen:

Threading

since Haxe 4.0.0

A unified threading API is available on some sys targets. The compile-time define `target.threaded` is set when the API is available. The API allows very simple creation of threads from functions:

```
class Main {
  public static function main():Void {
    #if (target.threaded)
    sys.thread.Thread.create(() -> {
      while (true) {
        trace("other thread");
        Sys.sleep(1);
      }
    });
    Sys.sleep(3);
    #end
  }
}
```

Asi creamos un Thread (**Sys.thread.Thread.create**), estos comparten todas las variables y memoria del programa principal, además de ejecutarse como deamons haciendo que por defecto el programa principal no espere por ellos. Para eso es necesario utilizar un Lock. Creamos una variable con las propiedades de lock y necesitamos llamar Lock.wait() por cada thread que queramos esperar. Y dentro de cada thread es necesario llamar a lock.release() para poder decirle al programa principal que puede proseguir.

Como se puede ver a la imagen, el thread ejecuta todo lo que esta en su espacio (Entre los parentesis), es decir que podemos definir con mucha exactitud que queremos que haga nuestro thread. Lo cual de cierta manera es muy convenite, desconozco si se le puede llamar con una función nada más, pero si puedes llamar a funciones definidas en cualquier lugar como si del main se refiriera. Para sincronización entre threads tenemos los mutex los cuales funcionan como siempre. Un hilo puede tener un mutex y los demás deben esperar a que este se libere. El programa principal no espera a que el mutex sea liberado, por lo que se debe jugar con los locks y con los mutex. Con respecto a los Process no conseguí mucha información al respecto, se que se pueden definir más no entendí a fondo como se hace.

En mi experiencia codeando en este lenguaje, pienso que es un lenguaje con mucho potencial que de verdad es cómodo de programar y es bastante intuitivo. Posee Iteradores y en conjunto a estos unas maneras muy interesantes de generar programas estilo Haskell, si bien no funcionaran exactamente igual tiene un estilo de diseño bastante funcional (al menos de manera visual).

6.4.2
Enum matching

Enums can be matched by their constructors in a natural way:

```
var myTree = Node(Leaf("foo"), Node(Leaf("bar"), Leaf("foobar")));
var match = switch (myTree) {
    // matches any Leaf
    case Leaf(_): "0";
    // matches any Node that has r = Leaf
    case Node(_, Leaf(_)): "1";
    // matches any Node that has
    // r = another Node, which has
    // l = Leaf("bar")
    case Node(_, Node(Leaf("bar"), _)): "2";
    // matches anything
    case _: "3";
  }
trace(match); // 2
```

The pattern matcher will check each case from top to bottom and pick the first one that matches the input value. The following manual interpretation of each case rule helps understanding the process:

- `case Leaf(_)`: matching fails because `myTree` is a `Node`
- `case Node(_, Leaf(_))`: matching fails because the right sub-tree of `myTree` is not a `Leaf`, but another `Node`
- `case Node(_, Node(Leaf("bar"), _))`: matching succeeds
- `case _`: this is not checked here because the previous line matched

Un pequeño inconveniente que me conseguí a la hora de realizar el reto especial, es que el sistema solo provee int32. Si bien no esta nada mal para el usuario promedio, puede que en algunos casos no sea suficiente. Existe un tipo de librería que ofrece unos int64 pero tiene poca documentación y no funciona de manera tan intuitiva. Le invito a revisar con calma el lenguaje, de verdad quedé encantado. Posee integración con al menos Visual Studio así que el IDE también aporta su ayuda para poder desarrollar código cómodo y legible. Sin más que decir esta es mi opinión sobre el lenguaje