Rockyll

“I feel like a Kentucky Fried Idiot”

- *Rocky Balboa*

Un Sung Yoo (uy2106)
Anish King (awk2123)

1. **Describe the language you plan to implement**
   1.1 Motivation

   Why do we always program in English?

   Spanish, Japanese, Korean, Tamil. These are a few of many native languages of some of the greatest programmers in the world. Our language attempts to successfully integrate “foreign” language speakers into a single programming language that successfully accommodates every talented programmer in the world.

   There are many talented programmers that would value a language that can compile their native mother tongue. Like Linux that gives the user the power, we are attempting to build a simple, minimalistic aesthetic that gives the user the power to customize and implement programs of their own design. We hope that the momentum of such a user-friendly language will promote an appreciation of open source development such that people around the world build things together that they may use together.

   1.2 Language Description

   Our language, very simply put, lets a user program in their native human language. That means for loops, if/else conditions, variables, and everything are implemented in the users’ own human language, converted via typeset and compiled effectively via OCaml, and output to C code / assembly code output to produce a versatile, beautiful, and all around robust programming language.

2. **Explain what sorts of programs are meant to be written in your language**

   In one word? ANYTHING.

   Our language is built for programmers that love imperative programming languages. Let’s think about the C language. C is a low-level imperative language that you can build an operating system with, as illustrated by the famous case of Linux and its many derivatives. Our language is a human-language friendly version of C. Think about programming in C, but you program in your mother tongue, be it Icelandic, French, or one of the many native languages of Papua New Guinea. Since we have only one semester for building this language, we will implement only a few human languages.

This presents a “limitation” but as previously described, we hope the momentum of our project will provide a platform for a significant level of open source development. We will implement, in a few select human languages known between us as the two programmers, basic imperative language necessities like creating functions, recursive definitions, and even object oriented programming basics.

Ultimately, we hope users of RockyII will go above and beyond what we, the creators, can possibly dream of to create systems that are robust, beautiful, and capable of translating human tasks into automated ones via one’s own mother tongue.

3. Explain the parts of your language and what they do

This language is based on established programming principles whilst being creative via an appreciation for our shared interest in a minimalistic aesthetic. See below.

Comments:

Single Line	//
Multi line	/* multi */

Data types:

int	Integers
float	Floating Values
char	Characters
bool	Boolean Value
string	Bunch of Characters
void	Or something like in ocaml a'

Data Structures:

The only included data structure is **arrays**.

The reason is other data structures can be built by the user, but arrays are necessary.

Operators, Comparators, Functions

Operators	+, -, *, / (we don’t offer other than these because it’s imperative!)

	++, --
Comparators	==, <=, >=, <, >, and (for English speakers, direct translation in other languages), or (for English speakers, direct translation in other languages)

Predefined Functions	print
	for printing, example:
	print “hello world”;
	enter
	for handling user input, example:
	int a;
	enter a; // this will be taken care as a′ (void in C)
	parseToChar(string, index)
	- parse string to char
	- strings will be built based on OCaml’s array function so they can parsed easily

Creating Functions	(data types) f(x) name {
	// statements
	}
	
	example: f(x) foo {
	int x = x;
	x = x + 1; // handling local variables
	return x;
	
	As in any imperative language, you may create any customized function that you so desire.

Basic Functions	main() {
	// statements
	}
	or
	main(string argv[]) {
	// statements
	}

Library option:

Use a customized library for OCaml and have it available for later use.
(Unsure? Please advise.)

4. Include the source code for an interesting program in your language

If we can write it in C, we can write it in Rockyll:

```
int f(int n) fib {
    if n < 2 {
        return n;
    else {
        fib(n - 1) + fib(n - 2); // our language can do recursive !!
    }
}

int main {
    if argv.size > 2 and argv.size < 2 {
    Err argv[0] + " [int values]\n";
}

    print fib(argv[1]);
    return 1;
}
```

Output

```
============================================================
./fib 3
2
./fib 3 2
Error Message: Supposed to be ./fib [int values]
./fib
Error Message: Supposed to be ./fib [int values]
============================================================
```

In Korean:

```
정수 함수(n) fib {                          // int f(n) fib
    조건 n < 2 {                            // if n < 2
        반납 n;                             // return n;
    } 그외 {                                // else
        반납 fib(n - 1) + fib(n - 2);       // return fib(n -1) + fib(n - 2);
    }
}

정수 본문 {                                 // int main
```

```
조건 인수.크기 > 2 그리고 인수.크기 < 2 {      // if argv.size > 2 and argv.size < 2
에러 인수[0] + "[int values]\n";                // Err argv[0] + “[int values]”;
}

출력 fib(인수[1]);                              // print fib(argv[1]);
반납 1;                                         return 1;
}
```

These two implementations of the same algorithm can be done in any human language thanks to RockyII.

This sample code shows but a snippet of the possibilities provided by RockyII.

5. Note.

We call our language RockyII because we are the underdogs. We are unexpected. We are heros. And moreover, RockyII is a hero, and it’s going to change the way programmers worldwide program.