SMAP

String Manipulation and Probability
Team

- Andrew
- Tushar
- Swapnil
- Emily
Purpose

Motivation: Procedural animation of games
C-like syntax, procedural, static type system
Support for dynamically sized arrays (lists) and probabilistic types (probs)
Winter Cabin Demo
A SMAP type as a list of types

```haskell
list list char = ["hello", "there"];
prob int x = [0.4, 0.4, 0.2] : [1,2,4];

type typ = Int | Bool | Float | Void | Char | String | Prob | List

type typ_name = typ list

type bind = typ_name * string
```
SMAP Goals

What if we could return user defined values according to some discrete probability distribution?

```plaintext
prob int x = [0.4, 0.4, 0.2] : [1,2,4];

print(x!); /* prints 1 40% of the time, 2 40% of the time, 4 20% of the time */
```
Smart Probs

Inputs are also normalized!

\[
\text{prob } \text{int } y = [0.1, 0.1] : [42, 7]; \quad // = [0.5, 0.5]
\]

Implemented probability transformations in C
Prob Highlights

- Probability

```java
int main()
{
    prob int num = [0.25, 0.5, 0.25] : [1, 2, 3];
    prob int num2 = num;
    int i;
    list float probs = num#;
    for(i=0; i < num.length; i = i+1)
    {
        println(probs[i]);
    }
}
```
List Highlights

- Lists

```java
int main()
{
    int z = 4;
    int w = 85;
    list int x = [0,1,2,3,z,99,7200,w-1];
    x[6] = (-7201);
    print(x);
}
```
Compiler Architecture
Built-in Functions

C library in SMAP for implementing prob type and lists. The library is imported through codegen which gives the SMAP language the flexibility to import the probabilistic feature.
Testing and built-in functions

- Many methods were built to support the implementation of Enigma machine to do string manipulation.
- 16 total methods
- Almost more than 75 tests which rigorously tests all the features including probability etc.
## Built-in methods

<table>
<thead>
<tr>
<th>Method</th>
<th>What it does</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>corresponding_int</td>
<td>Encode the input</td>
<td>Encode the input string to encrypt</td>
</tr>
<tr>
<td>corresponding_char</td>
<td>Decodes the input</td>
<td>Perform operation and revert the</td>
</tr>
<tr>
<td>key_test</td>
<td>Generate key sequence for testing</td>
<td>Picking random keys when performing cryptanalysis</td>
</tr>
<tr>
<td>ascii</td>
<td>Returns ascii of the character</td>
<td>Converts elements to their ascii values</td>
</tr>
<tr>
<td>int_to_char</td>
<td>Converts an integer to a character</td>
<td>Converts ascii to characters</td>
</tr>
<tr>
<td>ceilFloat</td>
<td>Generates the ceil value</td>
<td>Rounding off low probabilities</td>
</tr>
</tbody>
</table>
Enigma (Motivation)

- Build a complex machine that can extensively test and manipulate integers, lists and string.
- Used in WWII and is one of the most fascinating transposition cipher.
- Enigma machine was considered so secure that it was used to encipher the most top-secret messages.
- We wanted to simulate it.
- Making the process of testing fun.
- There is a movie about it.
Two types of Enigma
Prob Data type
Future work

- Lists and the probability feature together are highly adaptable to a gaming environment which was also our motivation.
- Using probability we can, in a controlled environment, randomize the characteristics of a payer inside a game.
- The probability feature can also randomize the attacks in a game to increase the surprise feature.
Demo

- Enigma Commercial
- Enigma Military
Output - Enigma

- Commercial Enigma Input:
  ZEROSIXHUNDREDDEEDEHOURSWEATHERTODAYISCLEARRAININTHEEVENINGHEILHLER

- Commercial Enigma output:
  WFORESTOSATXOFEGGETKLEKNUXZDOAXLHMCSAESQAEHAUSJAHQYUDSRUHSPDOXVPH

- Military Enigma Input:
  ZEROSIXHUNDREDDEEDEHOURSWEATHERTODAYISCLEARRAININTHEEVENINGHEILHLER

- Military Enigma output:
  WRORXFTOXIQXULKYQYZXCLPFXWMOAFJVUFFHOEAQHVAGLLAIQTFTTGOCCLZAMTXIIIH