SMAP

String Manipulation and Probability

Team

- Andrew
- Tushar
- Swapnil
- Emily

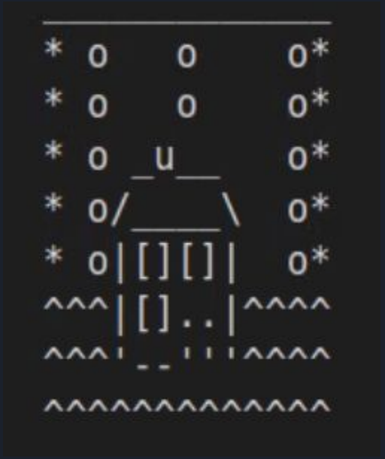
Purpose

Motivation: Procedural animation of games

C-like syntax, procedural, static type system

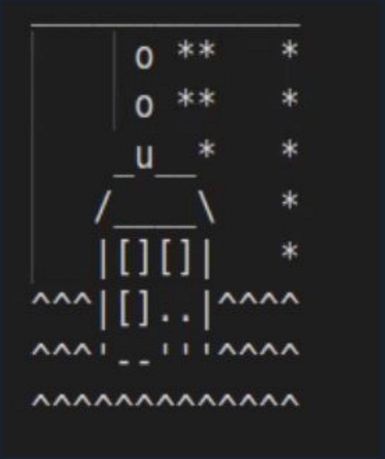
Support for dynamically sized arrays (lists) and probabilistic types (probs)

Winter Cabin Demo



```
_____________
* o   o    o*
* o   o    o*
* o _u__   o*
* o/____\  o*
* o|[][]|  o*
^^^|[]..|^^^^
^^^'__'''^^^^

^^^^^^^^^^^^^
```



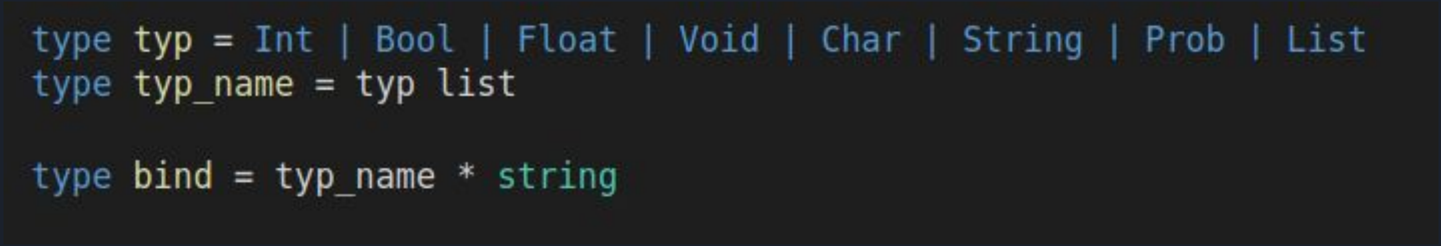
```
_____________
|   |o **   *
|   |o **   *
|   _u__*   *
|  /____\   *
|  |[][]|   *
^^^|[]..|^^^^
^^^'__'''^^^^

^^^^^^^^^^^^^
```

A SMAP type as a list of types

```
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?

```
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

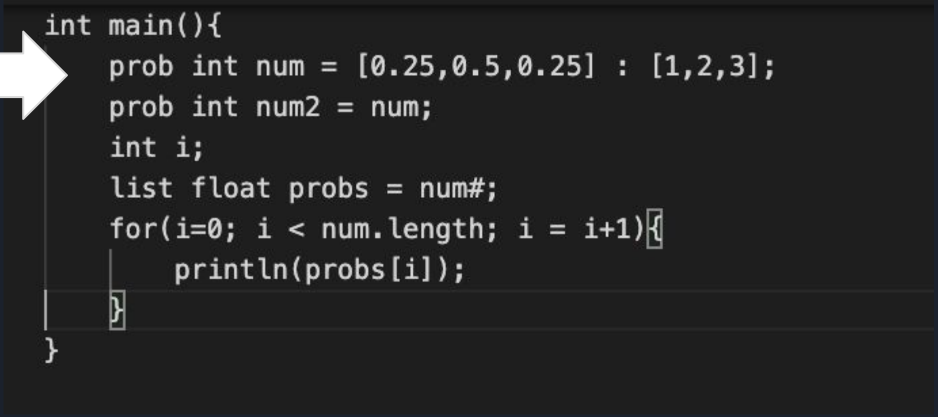
Inputs are also normalized!

```
prob int y = [0.1, 0.1] : [42, 7]; // =[0.5, 0.5]
```

`Implemented probability transformations in C`

Prob Highlights

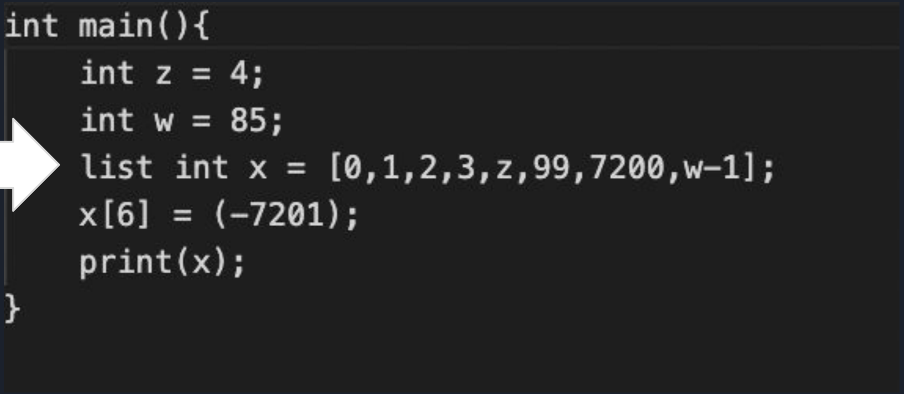
- Probability



```
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]);
    }
}
```

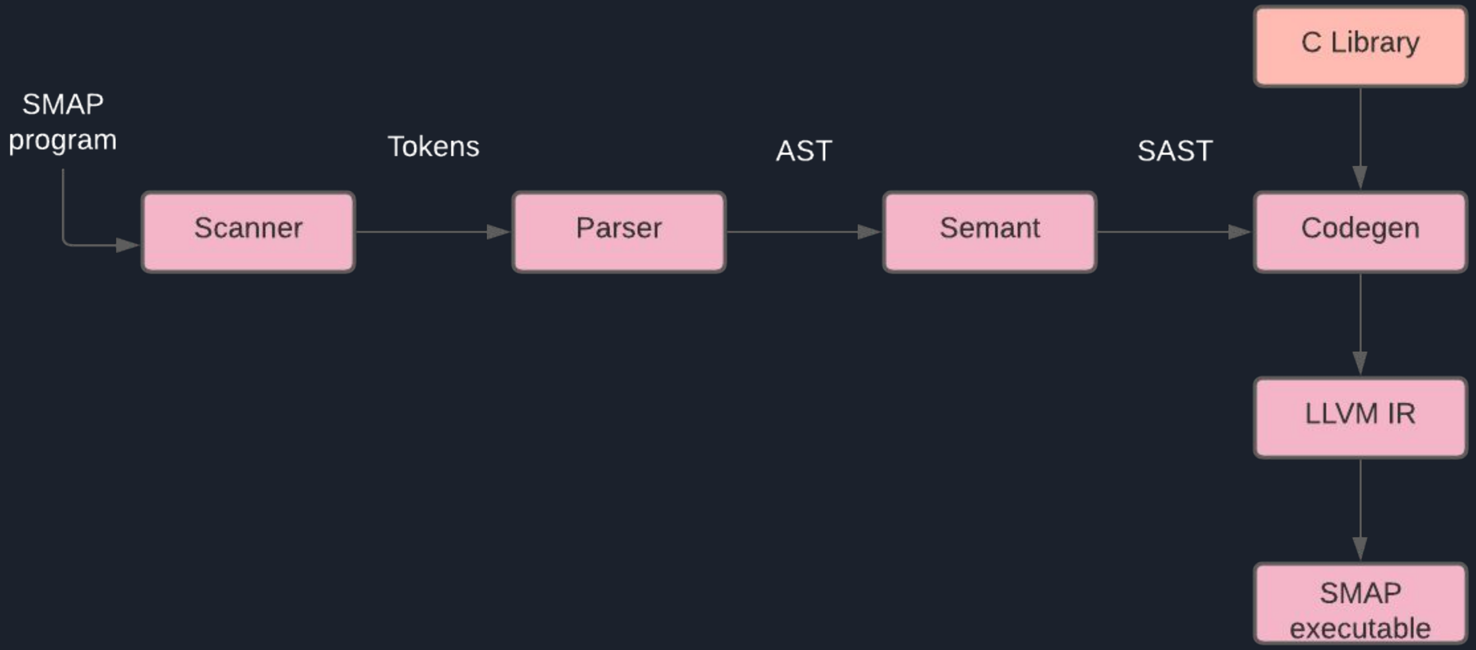
List Highlights

- Lists

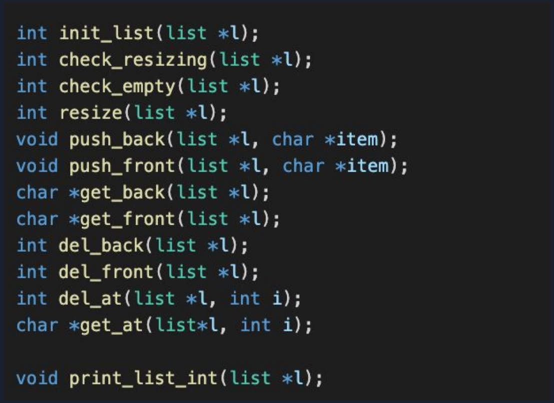


```
int main(){
    int z = 4;
    int w = 85;
    list int x = [0,1,2,3,z,99,7200,w-1];
    x[6] = (-7201);
    print(x);
}
```

Compiler Architecture



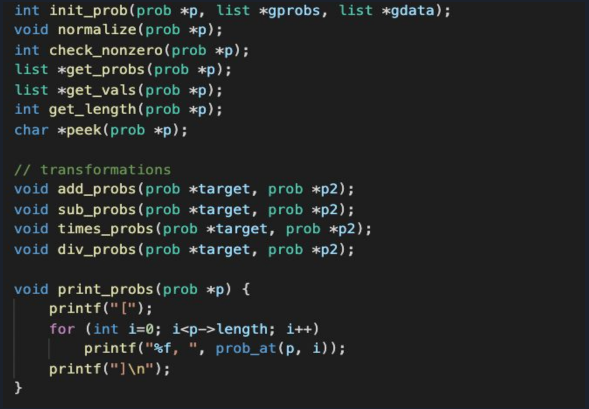
Built-in Functions



```
int init_list(list *l);
int check_resizing(list *l);
int check_empty(list *l);
int resize(list *l);
void push_back(list *l, char *item);
void push_front(list *l, char *item);
char *get_back(list *l);
char *get_front(list *l);
int del_back(list *l);
int del_front(list *l);
int del_at(list *l, int i);
char *get_at(list*l, int i);

void print_list_int(list *l);
```

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.



```
int init_prob(prob *p, list *gprobs, list *gdata);
void normalize(prob *p);
int check_nonzero(prob *p);
list *get_probs(prob *p);
list *get_vals(prob *p);
int get_length(prob *p);
char *peek(prob *p);

// transformations
void add_probs(prob *target, prob *p2);
void sub_probs(prob *target, prob *p2);
void times_probs(prob *target, prob *p2);
void div_probs(prob *target, prob *p2);

void print_probs(prob *p) {
    printf("[");
    for (int i=0; i<p->length; i++)
        printf("%f, ", prob_at(p, i));
    printf("]\n");
}
```

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

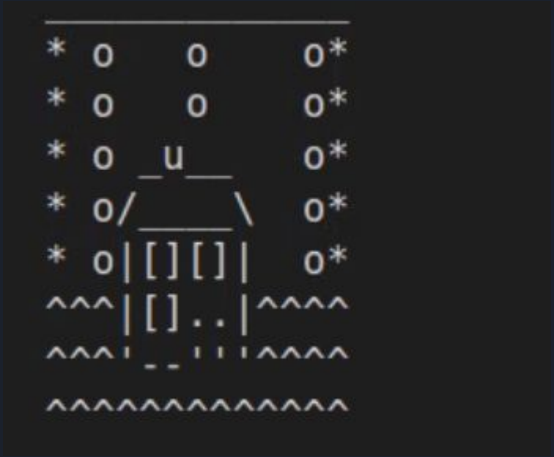
Method	What it does	Motivation
corresponding_int	Encode the input	Encode the input string to encrypt
corresponding_char	Decodes the input	Perform operation and revert the
key_test	Generate key sequence for testing	Picking random keys when performing cryptanalysis
ascii	Returns ascii of the character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

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.

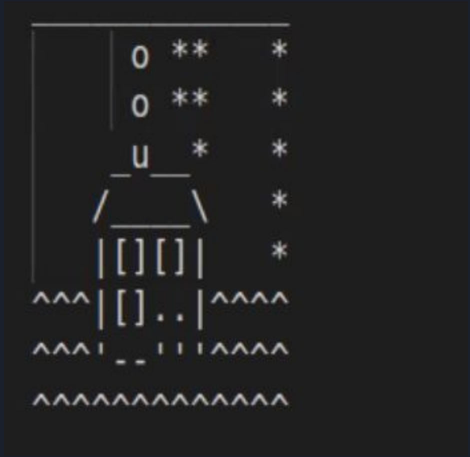
Two types of Enigma

Prob Data type



```
______________
* o   o    o*
* o   o    o*
* o _u__   o*
* o/____\  o*
* o|[][]|  o*
^^^|[]..|^^^^
^^^|__|||^^^^

^^^^^^^^^^^^^
```



```
_____________
|   | o **  *
|   | o **  *
|    _u__*  *
|   /____\  *
|   |[][]|  *
^^^|[]..|^^^^
^^^|__|||^^^^

^^^^^^^^^^^^^
```

Future work

- Lists and the probability feature together are highly adaptable to a gaming environment which was also out 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: ZEROSIXHUNDREEDHOURSWEATHERTODAYISCLEARRAININTHEEVENINGHEILHITLER
- Commercial Enigma output: WFORESTOSATXOFEGETKLEKNUXZDOAXLHMCSESQAEHAUUSJAHQUYDSRUIHSPDOXVPH
- Military Enigma Input: ZEROSIXHUNDREEDHOURSWEATHERTODAYISCLEARRAININTHEEVENINGHEILHITLER
- Military Enigma output: WRORXFTOXIQXULKYQYZXCLPFXWMOAFJVUFFHOEAQHVAGLLAIQTFTTGOCLZAMTXIIH