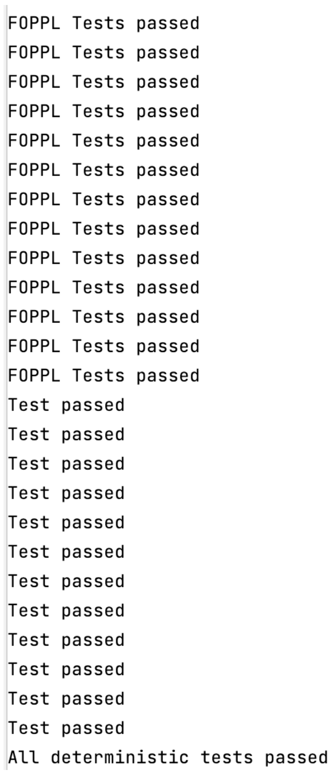
CPSC 532W - Homework 5

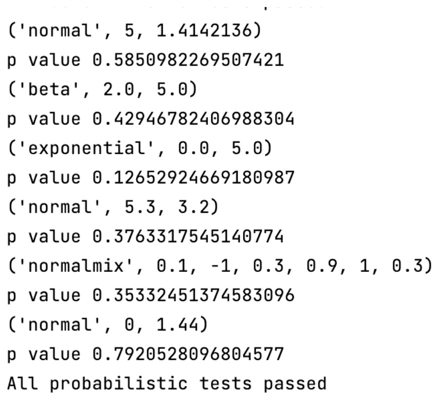
Xiaoxuan Liang - 48131163

Public GitHub repo: https://github.com/Xiaoxuan1121/CPSC532W/tree/main/a5

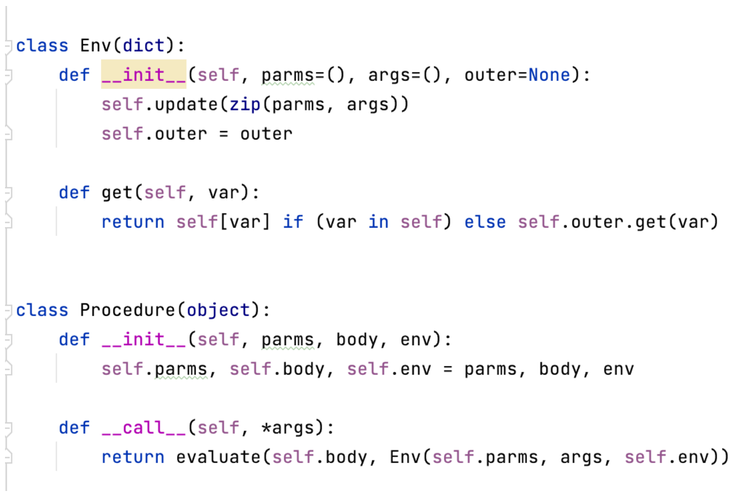
1. Program 1:



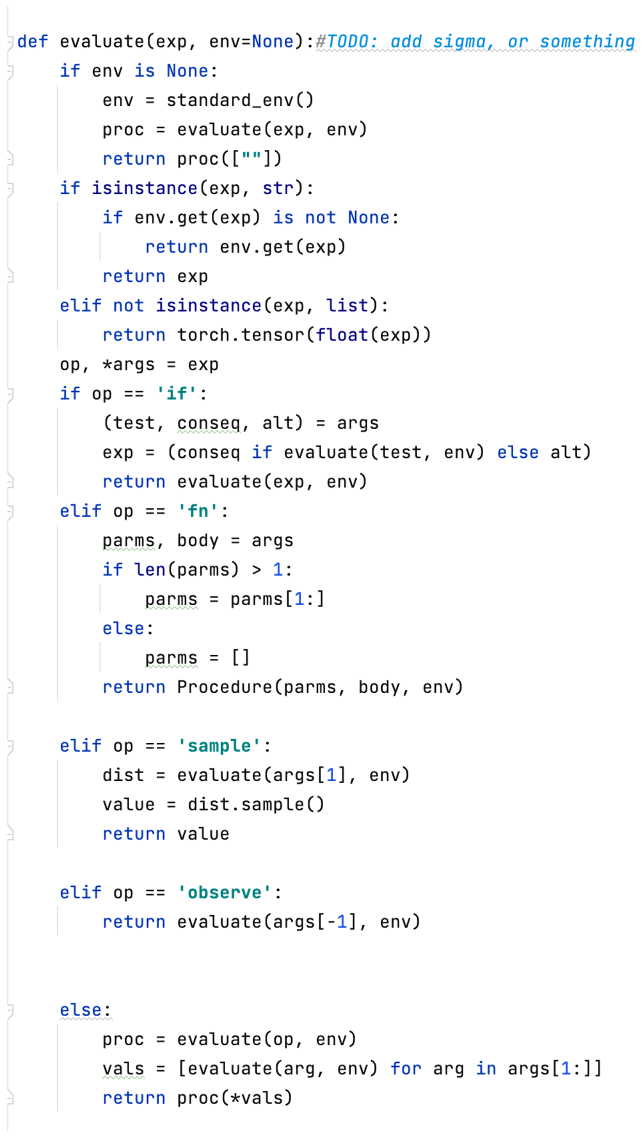
```
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
FOPPL Tests passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
Test passed
All deterministic tests passed
```



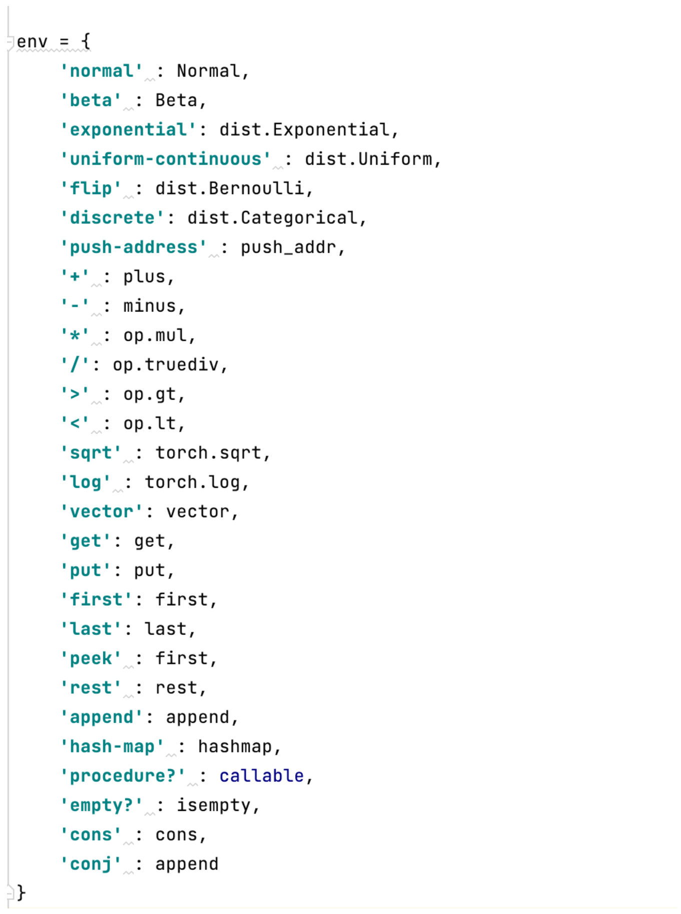
```
('normal', 5, 1.4142136)
p value 0.5850982269507421
('beta', 2.0, 5.0)
p value 0.42946782406988304
('exponential', 0.0, 5.0)
p value 0.12652924669180987
('normal', 5.3, 3.2)
p value 0.3763317545140774
('normalmix', 0.1, -1, 0.3, 0.9, 1, 0.3)
p value 0.35332451374583096
('normal', 0, 1.44)
p value 0.7920528096804577
All probabilistic tests passed
```



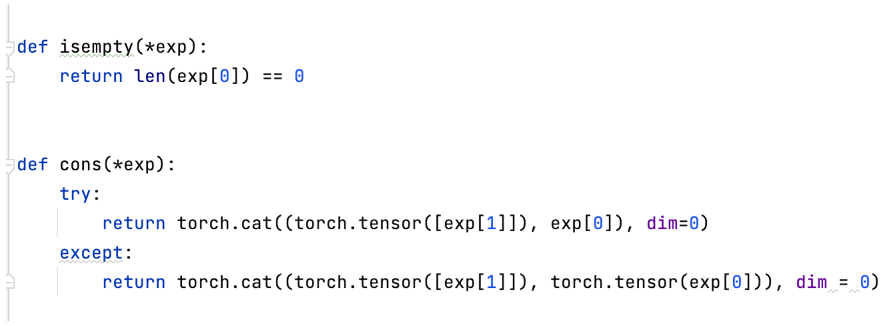
```python
class Env(dict):
    def __init__(self, parms=(), args=(), outer=None):
        self.update(zip(parms, args))
        self.outer = outer

    def get(self, var):
        return self[var] if (var in self) else self.outer.get(var)


class Procedure(object):
    def __init__(self, parms, body, env):
        self.parms, self.body, self.env = parms, body, env

    def __call__(self, *args):
        return evaluate(self.body, Env(self.parms, args, self.env))
```



```python
def evaluate(exp, env=None):#TODO: add sigma, or something
    if env is None:
        env = standard_env()
        proc = evaluate(exp, env)
        return proc([""])
    if isinstance(exp, str):
        if env.get(exp) is not None:
            return env.get(exp)
        return exp
    elif not isinstance(exp, list):
        return torch.tensor(float(exp))
    op, *args = exp
    if op == 'if':
        (test, conseq, alt) = args
        exp = (conseq if evaluate(test, env) else alt)
        return evaluate(exp, env)
    elif op == 'fn':
        parms, body = args
        if len(parms) > 1:
            parms = parms[1:]
        else:
            parms = []
        return Procedure(parms, body, env)

    elif op == 'sample':
        dist = evaluate(args[1], env)
        value = dist.sample()
        return value

    elif op == 'observe':
        return evaluate(args[-1], env)


    else:
        proc = evaluate(op, env)
        vals = [evaluate(arg, env) for arg in args[1:]]
        return proc(*vals)
```



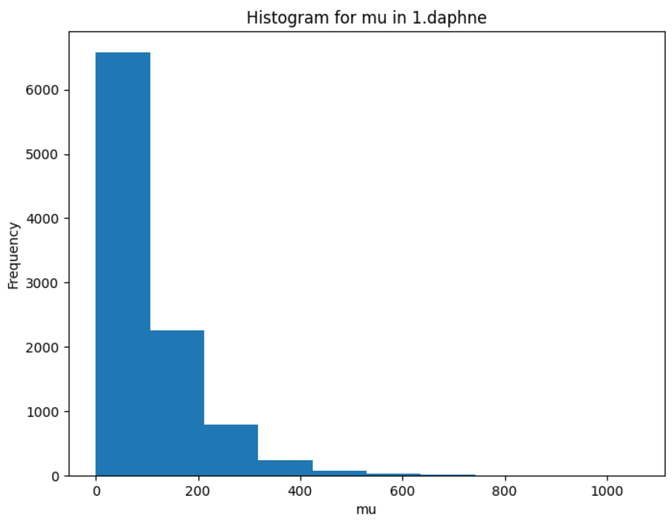
```python
env = {
    'normal' : Normal,
    'beta' : Beta,
    'exponential': dist.Exponential,
    'uniform-continuous' : dist.Uniform,
    'flip' : dist.Bernoulli,
    'discrete': dist.Categorical,
    'push-address' : push_addr,
    '+' : plus,
    '-' : minus,
    '*' : op.mul,
    '/': op.truediv,
    '>' : op.gt,
    '<' : op.lt,
    'sqrt' : torch.sqrt,
    'log' : torch.log,
    'vector': vector,
    'get': get,
    'put': put,
    'first': first,
    'last': last,
    'peek' : first,
    'rest' : rest,
    'append': append,
    'hash-map' : hashmap,
    'procedure?' : callable,
    'empty?' : isempty,
    'cons' : cons,
    'conj' : append
}
```



```python
def isempty(*exp):
    return len(exp[0]) == 0


def cons(*exp):
    try:
        return torch.cat((torch.tensor([exp[1]]), exp[0]), dim=0)
    except:
        return torch.cat((torch.tensor([exp[1]]), torch.tensor(exp[0])), dim = 0)
```

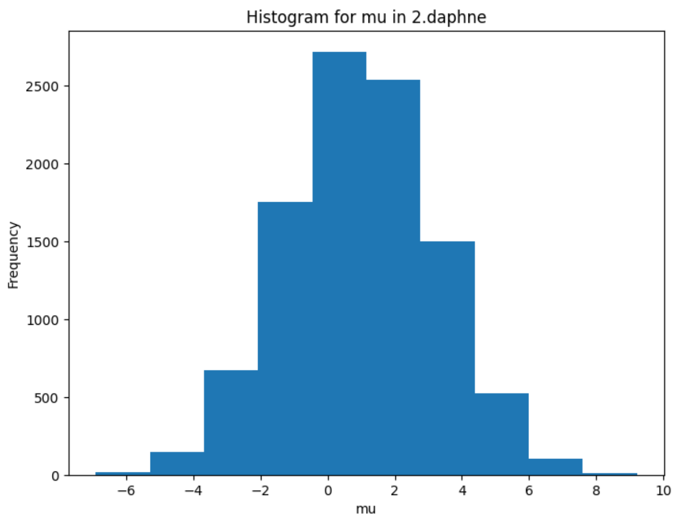
2. Program 2:

```
running 1.daphne took 92.527643 seconds

Prior expectation of mu in 1.daphne is 97.88899993896484

Prior variance of mu in 1.daphne is 9446.9794921875
```



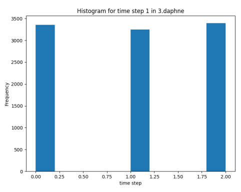
3. Program 3:

```
running 2.daphne took 4.908595 seconds

Prior expectation of mu in 1.daphne is 0.9962315559387207

Prior variance of mu in 1.daphne is 4.93194580078125
```

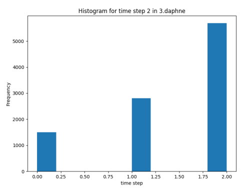


4. Program 4:

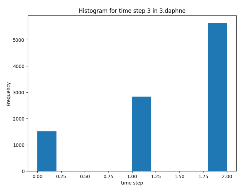
```
running 3.daphne took 26.665322 seconds
```



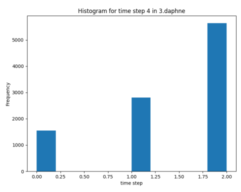
(a) Samples from the prior for HMM step 1



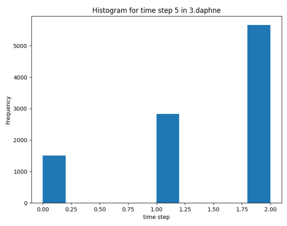
(b) Samples from the prior for HMM step 2



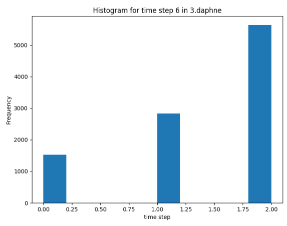
(c) Samples from the prior for HMM step 3



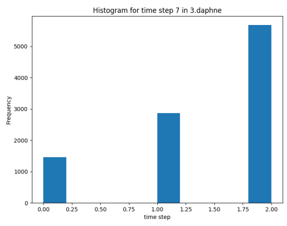
(d) Samples from the prior for HMM step 4



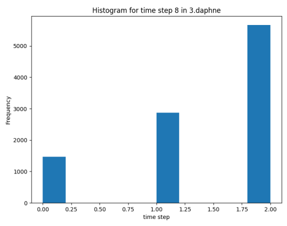
(e) Samples from the prior for HMM step 5



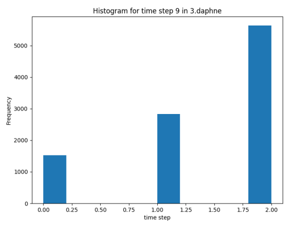
(f) Samples from the prior for HMM step 6



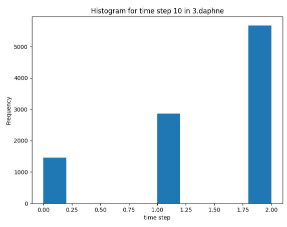
(g) Samples from the prior for HMM step 7



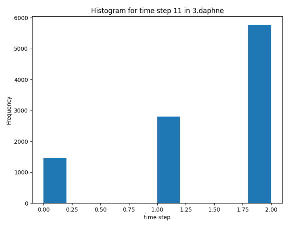
(h) Samples from the prior for HMM step 8



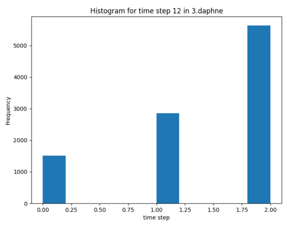
(i) Samples from the prior for HMM step 9



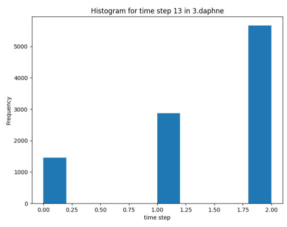
(j) Samples from the prior for HMM step 10



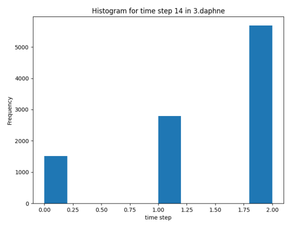
(k) Samples from the prior for HMM step 11



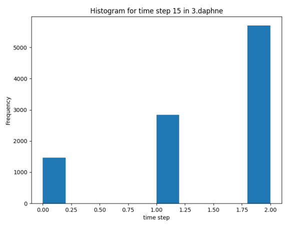
(l) Samples from the prior for HMM step 12



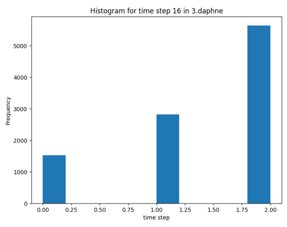
(m) Samples from the prior for HMM step 13



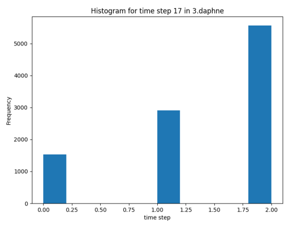
(n) Samples from the prior for HMM step 14



(o) Samples from the prior for HMM step 15



(p) Samples from the prior for HMM step 16



(q) Samples from the prior for HMM step 17

Figure 1: Evaluation: Partial Histograms for 3.daphne