```
double prob, path_prob = 1, choice_prob = 1;
int sum = 0, n = 0;

klee_make_symbolic(&prob, f"prob_sym_{i}");
klee_make_symbolic(&n, f"n_symbolic");
for i in range(n):
        int d;
        klee_make_symbolic(&d, f"d_sym_{i}");

        d = bernoulli(prob);
        (d == 1) ? choice_prob = prob : choice_prob = (1 - prob);
        path_prob = path_prob * choice_prob;

        sum = sum + d;

klee_dump(path_prob)
klee_dump(sum) // E[heads in "n" runs]
```

Fig. 1. Program listing for n bernoulli trails experiment.

We consider a independent *bernoulli* trials here of flipping a fair coin "n" times.

$$choice\_prob = \begin{cases} p & \text{if } d \text{ value is 1 corresponding to getting a "heads"} \\ 1-p & \text{if } d \text{ value is 0 corresponding to getting a "tails"} \end{cases}$$

concretely, on the i^{th} run $\vec{d}$ can have a value as below, one-hot encoded $w.r.t$ the outcome of *heads* or *tails*.

$$\vec{d_i} = encode(< 0, 0, 0, 1, 1, 0, 1, 1, 1, 0 >)$$

Based on the value of the $\vec{d_i}$, we get w_i value using $choice\_prob$.

$$w_i = (p)^{x_i} * (1-p)^{n-x_i} \tag{1}$$

where x_i denotes the number of *heads* in the i^{th} randomized run and for n runs. `n = 10` for the case in the above example.

$$w_i = (p)^{x_i} * (1-p)^{10-x_i} \tag{2}$$

We consider top "k" randomized runs now for the optimization query. The expression for `optimization` thus becomes

$$maximize(\sum_{i=1}^{k} w_i)$$

On substituting the value of w_i from (1).

$$maximize(\sum_{i=1}^{k} (p)^{x_i} * (1-p)^{n-x_i}) \tag{3}$$

After performing the optimization above, we get different values of $\vec{d_i}$. For i^{th} randomized run. we get a `single` one hot encoded $\vec{d}$ vector. We show below the encoding for a few i values.

$$\vec{d_1} = encode(< 0, 0, 0, 1, 1, 0, 1, 1, 1, 0 >)$$

$$Objective_1 = maximize(\sum_{i=1}^{k}(p)^{x_i} * (1-p)^{n-x_i}) \quad | \; \forall(i,j) \; [x_i \neq x_j] \tag{4}$$

Fig. 2. Optimization Expression for k randomized paths

$$\vec{d_3} = encode(< 0, 1, 0, 1, 1, 0, 0, 0, 1, 0 >)$$
$$\vec{d_4} = encode(< 1, 0, 1, 0, 1, 0, 1, 1, 0, 1 >)$$
$$....$$

We run the optimization by renaming the k `pse` variables set appropriately and then impose the *distinct* clause so that we don't run the optimization on the same *randomized* runs again.

$$\forall(i,j) \; [\vec{d_i} \neq \vec{d_j}] \tag{5}$$

We approximate the value of *expected* heads in the above program, with the following equations.

$$w_i = \prod_{j=1}^{n} choice\_prob_i(j), \quad sum_i = \sum_{j=1}^{n} components(\vec{d_i}), \tag{6}$$

$$EV(heads) = (\sum_{i=1}^{k} w_i * sum_i), \quad Error = n * prob - EV(heads) \tag{7}$$

where both w_i and sum_i can both be computed from the corresponding $\vec{d_i}$ expression we get from the *model* of the *optimization* query Eq 2

For $k = 5$ & $n = 10$ the two `constraint` sets and `optimization` expressions are as follows :

$$w_1 = \prod_{j=1}^{10} choice\_prob_1(j), \quad sum_1 = \sum_{j=1}^{10} components(\vec{d_1}), \tag{8}$$

$$w_2 = \prod_{j=1}^{10} choice\_prob_2(j), \quad sum_2 = \sum_{j=1}^{10} components(\vec{d_2}), \tag{9}$$

$$w_3 = \prod_{j=1}^{10} choice\_prob_3(j), \quad sum_3 = \sum_{j=1}^{10} components(\vec{d_3}), \tag{10}$$

$$w_4 = \prod_{j=1}^{10} choice\_prob_4(j), \quad sum_4 = \sum_{j=1}^{10} components(\vec{d_4}), \tag{11}$$

$$w_5 = \prod_{j=1}^{10} choice\_prob_5(j), \quad sum_5 = \sum_{j=1}^{10} components(\vec{d_5}), \tag{12}$$

$$\vec{d_1} \neq \vec{d_2} \neq \vec{d_3} \neq \vec{d_4} \neq \vec{d_5} \tag{13}$$

$$EV(heads) = (\sum_{i=1}^{k} w_i * sum_i), \tag{14}$$

$$EV(heads) = (w_1 * sum_1 \; + \; w_2 * sum_2 \; + \; w_3 * sum_3 \; + \; w_4 * sum_4 \; + \; w_5 * sum_5) \tag{15}$$

$$Objective_2 = maximize(w_1 * sum_1 \; + \; w_2 * sum_2 \; + \; w_3 * sum_3 \; + \; w_4 * sum_4 \; + \; w_5 * sum_5) \tag{16}$$

Statistically, maximizing $Objective_1$ must also maximize $Objective_2$ and thus we make the query $maximize(\sum_{i=1}^{k} w_i)$ to the optimizing solver finally.