応募課題のスクリーニングとランダム選択の手順

1　申請書を受理した順番で，1 から始まり 1 ずつ増加するエントリー番号 1, 2, 3, …, N をすべての申請書に付与する。

- 期間内に再送信した場合や，提出後取り下げた場合についても，最初に申請書を提出したタイミングでエントリー番号を付与する。
- エントリー番号は，受理あるいは募集締切りの時点で申請者に通知される。

2　募集要項に基づいて，申請書のスクリーニングを WG により行う。

3　スクリーニングを通過した課題から，添付の Python スクリプト（Python 3.7）を使用して 20 件をランダム選択で絞り込む。なお，スクリーニングを通過した課題数が 20 件を大きく超えない場合は，金額を調整のうえ全件を採択する場合がある。

3.1　ビットコインブロックチェーンにおける，6 月 26 日午前 8 時 0 分（日本時間）以降で一番早い順に 5 個のブロックのブロックハッシュの和を乱数シード S とする。random.seed(S)により乱数を初期化。

3.2　random.randint(1,N) によって $1 \leqq n \leqq N$ となる整数 n を得る。

3.3　得られた番号 n がスクリーニングを通過した申請書のエントリー番号に一致し，かつ，まだ採択されていなければ，その番号の課題を採択する。それ以外の場合（スクリーニングを通過しなかった課題，応募後に取り下げられた課題，すでに採択された課題）は何もしない。

3.4　20 課題が採択されるまで 3.2-3.3 を繰り返す。

4　採択課題の決定・通知時に，2 のスクリーニングを通過したエントリー番号の一覧と，3.1 で使用した乱数シードは公開される。

【補足】

1　ビットコインのブロックは約 10 分おきに新しく生成される。ブロック生成のたびに block height は 1 ずつ増加し，ブロック固有のハッシュ値（32 バイトの数値）が決まる。ブロックハッシュ値は以下のような性質を有するため，ランダム選択の乱数シードとして適している。

- 将来生成されるブロックハッシュを知ったり，望みの値に設定したりすることが，非常に困難（数百万円の費用がかかるため，実質上不可能）
  → ランダム選択の結果を締切り前に予想したり，不正に操作したりすることが（実質上）不可能
- 一度ブロックハッシュが決まれば，その値を誰でも知ることが可能
  → ランダム選択のプロセスに不正や誤りが無いことを，ブロックハッシュ値から計算した乱数シードを用いて，誰でも後から検証可能

2　添付の Python スクリプトでは，「ビットコインの block height 629530-629534 の 5 個のブロックハッシュの和」を乱数シードとした例を示している。

- ブロックハッシュ値は，https://www.blockchain.com/btc/block/[block height] から取得可能である。（例えば，block height 629530 のブロックハッシュであれば https://www.blockchain.com/btc/block/629530）
- Python スクリプトを実行すると下記の結果が出力される。乱数シードが同じ限り，何度実行しても同じ結果が得られる。

  Selected: [3, 4, 5, 6, 7, 8, 9, 10, 13, 14, 15, 17, 18, 19, 21, 23, 28, 29, 31, 33]

  Not selected: [1, 2, 11, 16, 20, 25]

ランダム選択を行う Python コード

https://ideone.com/tgi6lF

```
import random
import platform

# The maximum entry number
N = 35
# The number of selections
NUM_SELECTED = 20
# Entry numbers that passed the screening process (example is shown)
ENTRIES = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 25, 28, 29, 31,
33]
assert NUM_SELECTED < len(ENTRIES), "Selection must happen"

# Block hashes from certain block heights that were previously announced:
# Below is the example by block heights 629530-629534.
hashes = [
    0x00000000000000000006F349AA480F67A2B603496DA07FD0F566680293B2D3E4,
    0x0000000000000000000E4BF1CA971D88B29D31B84751AE6BDF8F2F5F25E5D99E,
    0x00000000000000000003A91B8D6D37940269AE8DE9219176DCD6BA448CE0AC75,
    0x0000000000000000000137A2AC232E19D2163A4A28B2F1F49CCD35052579451E,
    0x00000000000000000008A17371C0F62112227C28B83DD88C5218CAD648484E7F,
]

PYTHON_VERSION = platform.python_version()
if PYTHON_VERSION[0:3] != "3.7":
    print("Python version 3.7 must be used.")
    exit(0)

seed = sum(hashes)
random.seed(seed)
print("Entries:", ENTRIES, "(%d in total)" % len(ENTRIES))
print("The maximum entry number, N:", N)
print("The number of entries to select:", NUM_SELECTED)
print("Random seed: %d" % seed)
print()
selected = set()
count = 0
while True:
    s = random.randint(1, N)
    if s not in selected and s in ENTRIES:
        count += 1
        print("#%d:\t%d" % (count, s))
        selected.add(s)
    if count == NUM_SELECTED:
        break

entries = set(ENTRIES)
```

```
not_selected = entries.difference(selected)

print("\nResult")
print("Selected:", sorted(list(selected)))
print("Not selected:", sorted(list(not_selected)))
```