Zadanie 3

Polega na użyciu testera Dieharder, który implementuje zbiór testów statystycznych Diehard.

Dieharder posiada zbiór zarówno generatorów, jak również testów, wykorzystujących różne podejścia. Nie ma bowiem jednoznacznego sprawdzenia, czy próbki są losowe. Możemy raczej sprawdzić, że nie są przewidywalne.

"*Zawiera losowo całą bibliotekę naukową Gnu (GSL) generatory liczb (rngs), a także szereg generatorów z biblioteki statystycznej R, źródła sprzętowe, takie jak losowe / dev / *, generatory kryptograficznej jakości „złotego standardu" (przydatne do trudniejszych testów i do celów porównania z nowymi generatorami), a także generatory wniesione przez użytkowników lub znalezione w literaturze w jednej wiązce, która może je synchronizować i poddawać różnym testom losowości. Testy te są w różny sposób zaczerpnięte z „Diehard battery of random number tests" George'a Marsaglii, pakietu testów statystycznych NIST, a także z innych źródeł, takich jak osobiste wynalazki, wkład użytkowników, inne (open source) zestawy testów lub literatura.*" ~ opis z manuala

Jest to program dostępny na Linuxa, więc dla osób korzystających z Windowsa zalecam włączenie maszyny wirtualnej. Pierwszym krokiem jest przygotowanie środowiska.

```
sudo apt-get update
sudo apt-get install dieharder
```

Następnie w projekcie w pakiecie resources znajduje się gotowy plik o nazwie inputFile.txt, jest to plik, na którym bedziemy pracować.

Uwaga: Plik inputFile.txt można wygenerować samodzielnie, używając klasy Zadanie3. Plik zawiera 32mln intów, dlatego aby nie tracić czasu, udostępniłam swoje rozwiązanie.

Celem ćwiczenia jest zaprezentowanie dostępności łatwo obsługiwalnych testów generatorów przydatnych w przyszłej pracy.

Najważniejszymi opcjami diehardera bedą:

-a runs **all the tests** with standard/default options to create a user-controllable report. To control the formatting of the report, see -D below. To control the power of the test (which uses default values for tsamples that cannot generally be varied and psamples which generally can) see -m below as a "multiplier" of the default number of psamples (used only in a -a run).

-d test number - **selects specific diehard test.**

-f filename - generators 201 or 202 permit either raw binary or formatted ASCII numbers to be read in from a file for testing. generator 200 reads in raw binary numbers from stdin. Note well: many tests with default parameters require a lot of rands! To see a sample of the (required) header for ASCII formatted input, run

```
dieharder -o -f example.input -t 10
```

and then examine the contents of example.input. Raw binary input reads 32 bit increments of the specified data stream.

-g generator number - selects a specific generator for testing. Using **-g -1** causes **all known** generators to be printed out to the display.

```
anita ~]$ dieharder -g -1
#==============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown           #
#==============================================================================#
#    Id Test Name           | Id Test Name           | Id Test Name           #
#==============================================================================#
|    000 borosh13           |001 cmrg                |002 coveyou             |
|    003 fishman18          |004 fishman20           |005 fishman2x           |
|    006 gfsr4              |007 knuthran            |008 knuthran2           |
|    009 knuthran2002       |010 lecuyer21           |011 minstd              |
|    012 mrg                |013 mt19937             |014 mt19937_1999        |
|    015 mt19937_1998       |016 r250                |017 ran0                |
|    018 ran1               |019 ran2                |020 ran3                |
|    021 rand               |022 rand48              |023 random128-bsd       |
|    024 random128-glibc2   |025 random128-libc5     |026 random256-bsd       |
|    027 random256-glibc2   |028 random256-libc5     |029 random32-bsd        |
|    030 random32-glibc2    |031 random32-libc5      |032 random64-bsd        |
|    033 random64-glibc2    |034 random64-libc5      |035 random8-bsd         |
|    036 random8-glibc2     |037 random8-libc5       |038 random-bsd          |
|    039 random-glibc2      |040 random-libc5        |041 randu               |
|    042 ranf               |043 ranlux              |044 ranlux389           |
|    045 ranlxd1            |046 ranlxd2             |047 ranlxs0             |
|    048 ranlxs1            |049 ranlxs2             |050 ranmar              |
|    051 slatec             |052 taus                |053 taus2               |
|    054 taus113            |055 transputer          |056 tt800               |
|    057 uni                |058 uni32               |059 vax                 |
|    060 waterman14         |061 zuf                 |                        |
#==============================================================================#
|    200 stdin_input_raw    |201 file_input_raw      |202 file_input          |
|    203 ca                 |204 uvag                |205 AES_OFB             |
|    206 Threefish_OFB      |207 XOR (supergenerator)|208 kiss                |
|    209 superkiss          |                        |                        |
#==============================================================================#
|    400 R_wichmann_hill    |401 R_marsaglia_multic. |402 R_super_duper       |
|    403 R_mersenne_twister |404 R_knuth_taocp       |405 R_knuth_taocp2      |
#==============================================================================#
|    500 /dev/random        |501 /dev/urandom        |                        |
#==============================================================================#
#==============================================================================#
```

-l lists **all the tests** available

```
anita ~]$ dieharder -l
#==============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown           #
#==============================================================================#
Installed dieharder tests:
 Test Number                          Test Name               Test Reliability
==============================================================================
  -d 0                          Diehard Birthdays Test             Good
  -d 1                             Diehard OPERM5 Test             Good
  -d 2                   Diehard 32x32 Binary Rank Test            Good
  -d 3                     Diehard 6x8 Binary Rank Test            Good
  -d 4                           Diehard Bitstream Test            Good
  -d 5                                    Diehard OPSO           Suspect
  -d 6                                    Diehard OQSO           Suspect
  -d 7                                     Diehard DNA           Suspect
  -d 8               Diehard Count the 1s (stream) Test             Good
  -d 9                 Diehard Count the 1s Test (byte)             Good
  -d 10                       Diehard Parking Lot Test             Good
  -d 11        Diehard Minimum Distance (2d Circle) Test            Good
  -d 12        Diehard 3d Sphere (Minimum Distance) Test            Good
  -d 13                         Diehard Squeeze Test               Good
  -d 14                            Diehard Sums Test           Do Not Use
  -d 15                            Diehard Runs Test               Good
  -d 16                          Diehard Craps Test                Good
  -d 17                   Marsaglia and Tsang GCD Test             Good
  -d 100                           STS Monobit Test               Good
  -d 101                              STS Runs Test               Good
  -d 102                  STS Serial Test (Generalized)            Good
  -d 200                      RGB Bit Distribution Test            Good
  -d 201          RGB Generalized Minimum Distance Test            Good
  -d 202                        RGB Permutations Test              Good
  -d 203                          RGB Lagged Sum Test              Good
  -d 204               RGB Kolmogorov-Smirnov Test Test            Good
  -d 205                            Byte Distribution              Good
  -d 206                                    DAB DCT                Good
  -d 207                          DAB Fill Tree Test               Good
  -d 208                        DAB Fill Tree 2 Test               Good
  -d 209                          DAB Monobit 2 Test               Good
```

Przykładowe użycie programu:

```
anita ~]$ dieharder -d 0 -g 501
#=============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================#
   rng_name    |rands/second|   Seed   |
   /dev/urandom|  7.80e+06  |  56656064|
#=============================================================================#
        test_name   |ntup| tsamples |psamples|  p-value |Assessment
#=============================================================================#
   diehard_birthdays|   0|       100|     100|0.93792762|  PASSED
```

Zadania do wykonania:

1) Włączyć program dieharder i uruchomić wszystkie testy (-a) na pliku inputFile.txt
Po wykonaniu testu o nazwie *diehard_craps* wyłączyć program wpisując Ctrl-C.

Jako odpowiedź podać: czy wszystkie testy zostały zdane? Jeśli nie, to które nie zostały.

2) Opisać, czym jest p-value – na co wskazuje, czy lepsza jest duża czy niewielka wartość, czy jednoznacznie określa, że liczby są losowe?

Uwaga: dla osób niesłuchających na prezentacji odpowiedź można uzyskać wpisując komendę *man dieharder*.

3) Opisać, czym są /dev/random i /dev/urandom oraz czym się różnią.

Ważne źródło:
https://webhome.phy.duke.edu/~rgb/General/dieharder.php