Chapitre 4 – Algorithmes mathématiques – Méthode MonteCarlo **Exercice Algo 4.2**
Mathematische Algorithmen – Monte Carlo Methode

Dans cet exercice nous voulons effectuer une simulation Monte Carlo pour déterminer la répartition des points des dés. Pour ce faire, on lance virtuellement x dés (x = [2....20] et calcule la somme des points de ces dés. En répétant ça plusieurs millions fois, on reçoit une répartition qui devrait correspondre à une répartition théorique.

In dieser Aufgabe wollen wir eine Monte Carlo Simulation durchführen, um die Verteilung der Augen von Würfeln zu bestimmen. Dazu werfen wir virtuell x Würfel und addieren die Summe ihrer Augen. In dem wir dies millionenfach wiederholen, erhalten wir eine Verteilung, die einer theoretischen Verteilung entsprechen müsste.

a.) Calculez manuellement la réparation des points pour 2 et 3 dés. (2 dés :valeurs entre 2 et 12 ; 3 dés : valeurs entre 3 et 18). Notez aussi les valeurs relatives (en%).

Berechnen Sie manuell die Verteilung der Augenzahlen für 2 und 3 Würfel. (2 Würfel ; Werte zwischen 2 und 12 ; für 3 Würfel : Werte zwischen 3 und 18). Schreiben Sie auch die relativen Werte (in%) auf.

Chaque possibilité de combinaisons de 2 dés a une probabilité de :
Jede mögliche Kombination von 2 Würfeln hat eine Wahrscheinlichkeit von :

1/36=2.78 %

2 : 2.8% : 1 (11)
3 : 5.6% : 2 (12,21)
4 : 8.3% : 3 (13,22,31)
5 : 11.1% : 4 (14,23,32,41)
6 : 13.8% : 5 (15,24,33,42,51)
7 : 16.6% : 6 (16,25,34,43,54,61)
8 : 13.8% : 5 (26,35,44,53,26)
9 : 11.1% : 4 (36,45,54,63)
10 : 8.3% : 3 (46,55,64)
11 : 5.6% : 2 (56,65)
12 : 2.8% : 1 (66)

Chaque possibilité de combinaisons de 3 dés a une probabilité de :
Jede mögliche Kombination von 3 Würfeln hat eine Wahrscheinlichkeit von :

1/(6*6*6)=1/216 ≈ 0.463%

3 : 0.46% : 1 (111)

4 : 1.39% : 3 (211, 121, 112)

5 : 2.78% : 6 (311, 221, 212, 122, 131, 113)

6 : 4.63% : 10 (411, 321, 312, 231, 222, 213, 141, 132, 123, 114) 4.63 %

7 : 6.94% : 15 (511, 421, 412, 331, 322, 313, 241, 232, 223, 214, 151, 142, 133, 124, 115)

8 : 9.72% : 21 (611, 521, 512, 431, 422, 413, 341, 332, 323, 314, 251, 242, 233, 224, 215, 161, 152, 143, 134, 125, 116)

9 : 11.57% : 25 (621, 612, 531, 522, 513, 441, 432, 423, 414, 351, 342, 333, 324, 315, 261,252, 243, 234, 225, 216, 162, 153, 144, 135, 126)

10 : 12.50% : 27 (631, 622, 613, 541, 532, 523, 514, 451, 442, 433, 424, 415, 361, 352, 343, 334, 325, 316, 262, 253, 244, 235, 226, 163, 154, 145, 136)

11 : 12.50% : 27 ...

12 : 11.57% : 25 ...

13 : 9.72% : 21 ...

14 : 6.94% : 15 ...

15 : 4.63% : 10 ...

16 : 2.78% : 6 ...

17 : 1.39% : 3 ...

18 : 0.46% : 1 ...

b.) Implémentez un programme qui calcule la répartition de chaque nombre de points possible. Le programme a le nom ***dices*** et prend 2 paramètres : ***number_of_dices*** : combien de dés veut-on jeter et ***number_of_throws*** : combien de jets veut-on faire. Le programme sort le nombre relative de chaque possibilité de nombre de points.

Implementieren Sie ein Programm, welches die Verteilung von allen möglichen Augenzahl berechnet. Das Programm hat den Namen ***dices*** und wird mit 2 Parametern aufgerufen : ***number_of_dices*** : wie viele Würfel man werfen möchte und ***number_of_throws*** : wie viele Würfe man machen möchte.

dices <number_of_dices> <number_of_throws>

```c
/**
  ******************************************************************************
  * @file    main.c
  * @author  Wolfram Luithardt / Roland Scherwey
  * @date    8 December 2019
  * @brief   Exercise 4.2 Monte Carlo algorithm for a game with dices
  ******************************************************************************/

#include <stdio.h>  // for printf()
#include <stdlib.h> // for srand() / rand() / RAND_MAX / exit()
#include <time.h>   // for time()
#include <string.h> // for memset()

int main(int argc, char *argv[]){
  if (argc !=3){
    printf("Programm needs 2 parameters!!!\n");
    printf("appname <number_of_dices> <number_of_throws>\n");
    exit(1);
  }

  unsigned int const cNumberOfDices = atoi(argv[1]);
  if(cNumberOfDices < 1  || cNumberOfDices > 20){
    printf("Number of dices must be between 1 and 20!\n");
    exit(2);
  }

  unsigned int cNumberOfThrows = atoi(argv[2]);
  if(cNumberOfThrows < 1 || cNumberOfThrows > 10000000){
    printf("Number of throws must be between 1 and 10000000!\n");
    exit(3);
  }

  printf("Entered parameters are: dices=%u and throws=%u  \n",
         cNumberOfDices, cNumberOfThrows);

  srand(time(NULL)); // seeds the random number generator used by rand()

  // Create array for all possible points (ignore element array[0] -> + 1)
  unsigned int array[cNumberOfDices*6+1];
  memset(array,0,sizeof (array));// init all elements to 0

  // Create necessary amount of throws
  for (unsigned int i=0; i<cNumberOfThrows; i++){
    unsigned int sum=0;
    for (unsigned int j=0; j<cNumberOfDices; j++){ // with each dice
        unsigned int const cNum = (unsigned int)((double)rand()/(RAND_MAX+1) *6.0) + 1;
        sum += cNum;
    }
    array[sum]++;
  }

  // Compute probability for each possible combination
  for (unsigned int i=cNumberOfDices; i<=cNumberOfDices*6; i++)
  {
     double const cPercent= (double) array[i]/cNumberOfThrows*100;
     printf("%2d  : %.3f%c \n", i, cPercent, '%');
  }

  return 0;
}
```