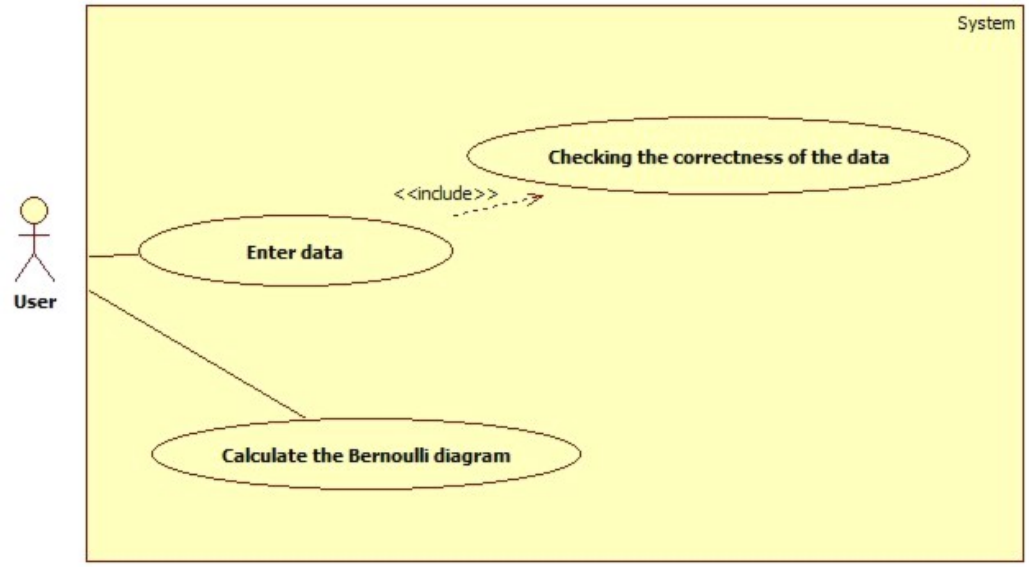
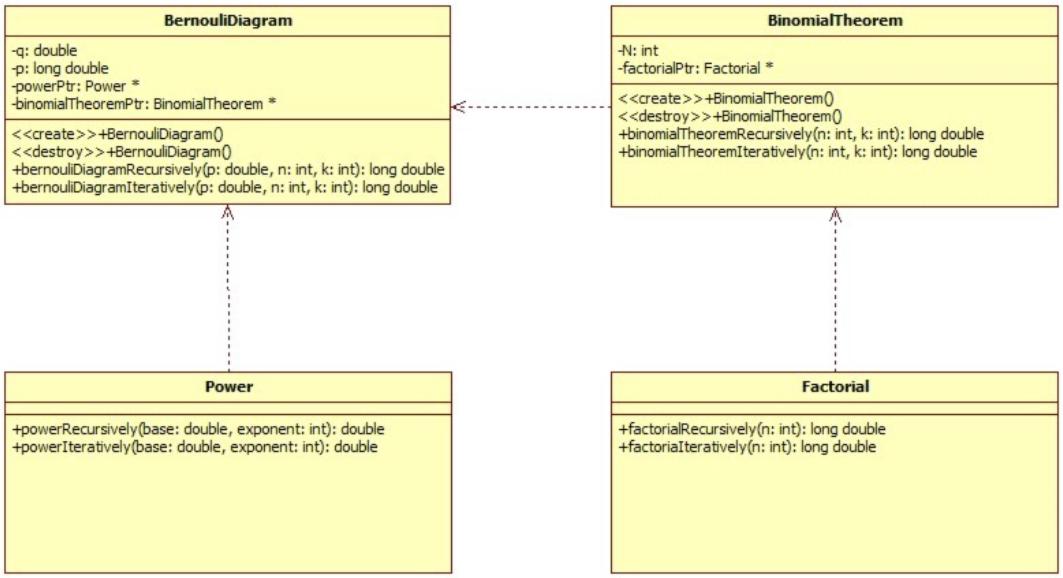
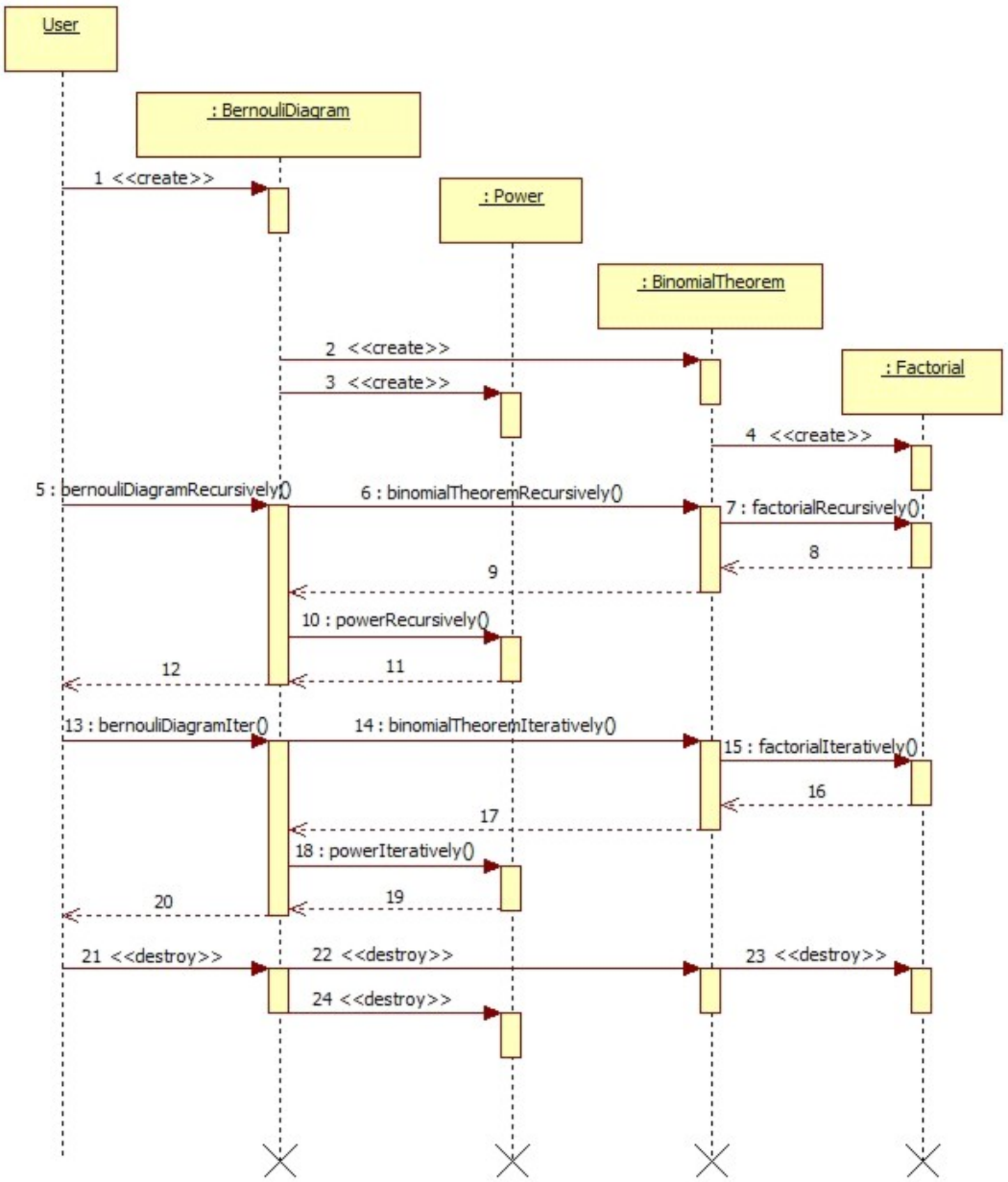

System
Checking the correctness of the data
<<include>>
Enter data
User
Calculate the Bernoulli diagram


BernouliDiagram
-q: double
-p: long double
-powerPtr: Power *
-binomialTheoremPtr: BinomialTheorem *
<<create>>+BernouliDiagram()
<<destroy>>+BernouliDiagram()
+bernouliDiagramRecursively(p: double, n: int, k: int): long double
+bernouliDiagramIteratively(p: double, n: int, k: int): long double
BinomialTheorem
-N: int
-factorialPtr: Factorial *
<<create>>+BinomialTheorem()
<<destroy>>+BinomialTheorem()
+binomialTheoremRecursively(n: int, k: int): long double
+binomialTheoremIteratively(n: int, k: int): long double
Power
+powerRecursively(base: double, exponent: int): double
+powerIteratively(base: double, exponent: int): double
Factorial
+factorialRecursively(n: int): long double
+factoriaIteratively(n: int): long double



//

//

// Generated by StarUML(tm) C++ Add-In

//

// @ Project : Laboratorium 08c

// @ File Name : BernouliDiagram.h

// @ Date : 31.05.2023

// @ Author : Tomasz Wnuk

//

```
//


#if !defined(_BERNOULIDIAGRAM_H)

#define _BERNOULIDIAGRAM_H


#include "BinomialTheorem.h"

#include "Power.h"


class BernouliDiagram {

public:

		BernouliDiagram();

		~BernouliDiagram();

		long double bernouliDiagramRecursively(double p, int n, int k);

   long double bernouliDiagramIteratively(double p, int n, int k);

private:

		double q;

		long double p;

		Power * powerPtr;

		BinomialTheorem * binomialTheoremPtr;

};


#endif  //_BERNOULIDIAGRAM_H

//

//

//  Generated by StarUML(tm) C++ Add-In

//

//  @ Project : Laboratorium 08c

//  @ File Name : BernouliDiagram.cpp

//  @ Date : 31.05.2023
```

```
//  @ Author : Tomasz Wnuk
//
//

#include <iostream>
#include "BernouliDiagram.h"

using namespace std;

BernouliDiagram::BernouliDiagram() {
   binomialTheoremPtr = new BinomialTheorem();
   powerPtr = new Power();
}

BernouliDiagram::~BernouliDiagram() {
   delete binomialTheoremPtr;
   delete powerPtr;
}

long double BernouliDiagram::bernouliDiagramRecursively(double p, int n, int k) {
   q = 1 - p;
   return ((binomialTheoremPtr->binomialTheoremRecursively(n, k)) * (powerPtr->powerRecursively(p, k))
* powerPtr->powerRecursively(q, n - k));
}

long double BernouliDiagram::bernouliDiagramIteratively(double p, int n, int k) {
   q = 1 - p;
   return ((binomialTheoremPtr->binomialTheoremIteratively(n, k)) * (powerPtr->powerIteratively(p, k)) *
powerPtr->powerIteratively(q, n - k));
```

```
}
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 08c
//  @ File Name : BinomialTheorem.h
//  @ Date : 31.05.2023
//  @ Author : Tomasz Wnuk
//
//

#if !defined(_BINOMIALTHEOREM_H)
#define _BINOMIALTHEOREM_H

#include "Factorial.h"

class BinomialTheorem {
public:
		BinomialTheorem();
		~BinomialTheorem();
		long double binomialTheoremRecursively(int n, int k);
		long double binomialTheoremIteratively(int n, int k);
private:
		int N;
		Factorial * factorialPtr;
};

#endif  //_BINOMIALTHEOREM_H
```

```
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 08c
//  @ File Name : BinomialTheorem.cpp
//  @ Date : 31.05.2023
//  @ Author : Tomasz Wnuk
//
//

#include "BinomialTheorem.h"

BinomialTheorem::BinomialTheorem() {
   factorialPtr = new Factorial();
}

BinomialTheorem::~BinomialTheorem() {
   delete factorialPtr;
}

long double BinomialTheorem::binomialTheoremRecursively(int n, int k) {
   N = n - k;
   long NbyK = 1;

   if(k >= N) {
      for(int i = k + 1; i <= n; i++) {
         NbyK *= i;
      }
```

```
        return (NbyK / factorialPtr->factorialRecursively(n - k));
    } else {
        for(int i = N + 1; i <= n; i++) {
            NbyK *= i;
        }
        return (NbyK / factorialPtr->factorialRecursively(k));
    }
}

long double BinomialTheorem::binomialTheoremIteratively(int n, int k) {
    N = n - k;
    long NbyK = 1;
    long long factorialNK = 1;
    long long factorialK = 1;

    if (k >= N) {
        for (int i = k + 1; i <= n; i++) {
            NbyK *= i;
        }
        for (int i = 1; i <= n - k; i++) {
            factorialNK *= i;
        }
        return (NbyK / factorialNK);
    } else {
        for (int i = N + 1; i <= n; i++) {
            NbyK *= i;
        }
        for (int i = 1; i <= k; i++) {
            factorialK *= i;
```

```
        }
        return (NbyK / factorialK);
    }
}
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 08c
//  @ File Name : Factorial.h
//  @ Date : 31.05.2023
//  @ Author : Tomasz Wnuk
//
//

#if !defined(_FACTORIAL_H)
#define _FACTORIAL_H

class Factorial {
public:
                long double factorialRecursively(int n);
                long double factorialIteratively(int n);
};

#endif  //_FACTORIAL_H
//
//
//  Generated by StarUML(tm) C++ Add-In
//
```

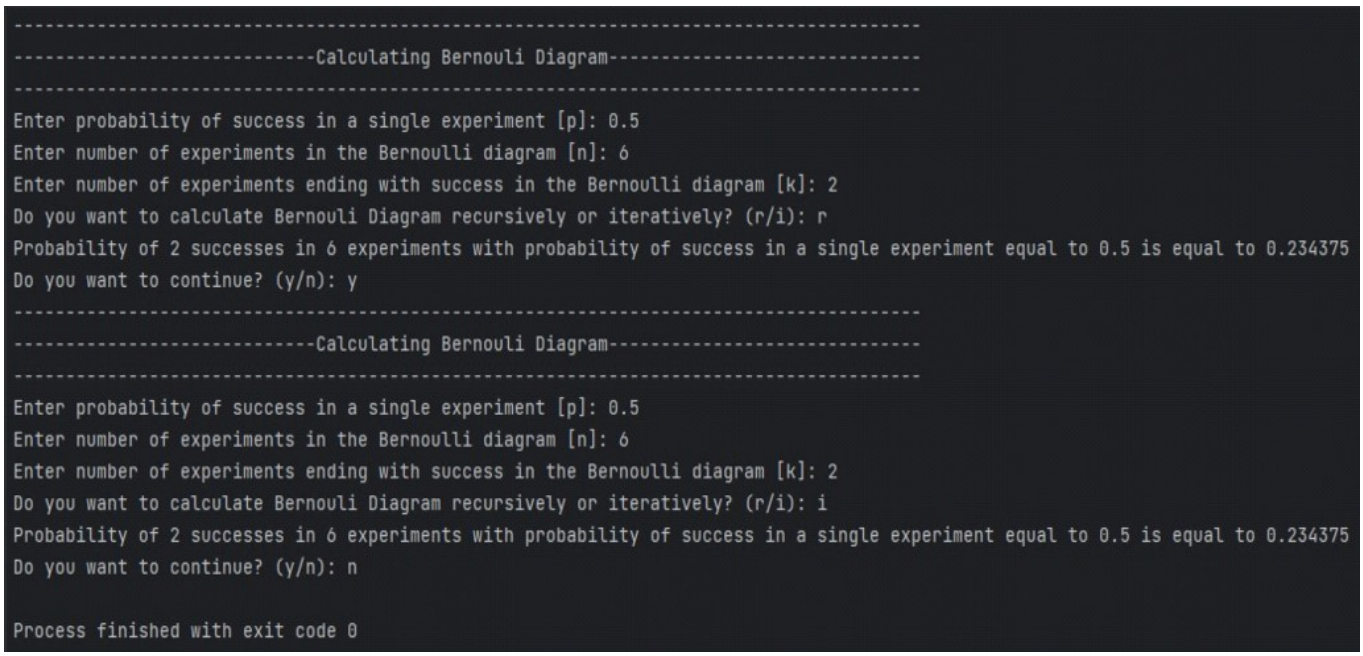
```
//  @ Project : Laboratorium 08c
//  @ File Name : Factorial.cpp
//  @ Date : 31.05.2023
//  @ Author : Tomasz Wnuk
//
//

#include "Factorial.h"

long double Factorial::factorialRecursively(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * factorialRecursively(n - 1);
    }
}

long double Factorial::factorialIteratively(int n) {
    long double result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 08c
```

```
//  @ File Name : Power.h
//  @ Date : 31.05.2023
//  @ Author : Tomasz Wnuk
//
//

#if !defined(_POWER_H)
#define _POWER_H

class Power {
public:
		double powerRecursively(double base, int exponent);
		double powerIteratively(double base, int exponent);
};

#endif  //_POWER_H
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 08c
//  @ File Name : Power.cpp
//  @ Date : 31.05.2023
//  @ Author : Tomasz Wnuk
//
//

#include "Power.h"
```

```
double Power::powerRecursively(double base, int exponent) {
   if(exponent == 0)
      return 1;
   else {
      return base * powerRecursively(base, exponent - 1);
   }
}

double Power::powerIteratively(double base, int exponent) {
   double result = 1;
   for(int i = 0; i < exponent; i++) {
      result *= base;
   }
   return result;
}
#include <iostream>
#include "BernouliDiagram.h"

using namespace std;

int main() {
   // Create BernouliDiagram
   BernouliDiagram * bernouliDiagram = new BernouliDiagram();

   // Infinite loop
   while(true) {
      // Probability of success in a single experiment
      double p;
      // Number of experiments in the Bernoulli diagram
```

```
int n;

// Number of experiments ending with success in the Bernoulli diagram

int k;


// Calculate probability with BernouliDiagram

cout << "-------------------------------------------------------------------------------------\n";

cout << "----------------------------Calculating Bernouli Diagram----------------------------\n";

cout << "-------------------------------------------------------------------------------------\n";


// Get user input

cout << "Enter probability of success in a single experiment [p]: ";

cin >> p;


cout << "Enter number of experiments in the Bernoulli diagram [n]: ";

cin >> n;


cout << "Enter number of experiments ending with success in the Bernoulli diagram [k]: ";

cin >> k;


// Get user input

string userInput;

cout << "Do you want to calculate Bernouli Diagram recursively or iteratively? (r/i): ";

cin >> userInput;


// Print result

if(userInput == "r") {

    cout << "Probability of " << k << " successes in " << n << " experiments with probability of success in a single experiment equal to " << p << " is equal to " << bernouliDiagram->bernouliDiagramRecursively(p, n, k) << endl;
```

```
    } else if(userInput == "i") {
        cout << "Probability of " << k << " successes in " << n << " experiments with probability of success in a single experiment equal to " << p << " is equal to " << bernouliDiagram->bernouliDiagramIteratively(p, n, k) << endl;
    } else {
        cout << "Wrong input" << endl;
    }

    cout << "Do you want to continue? (y/n): ";
    cin >> userInput;
    // Check if user wants to continue
    while(true) {
        // Check if user wants to continue
        if(userInput == "y") {
            break;
        } else if(userInput == "n") {
            delete bernouliDiagram;
            return 0;
        } else {
            cout << "Wrong input" << endl;
        }
        // Get user input
        cin >> userInput;
    }
  }

  return 0;
}
```



wyk. Tomasz Wnuk