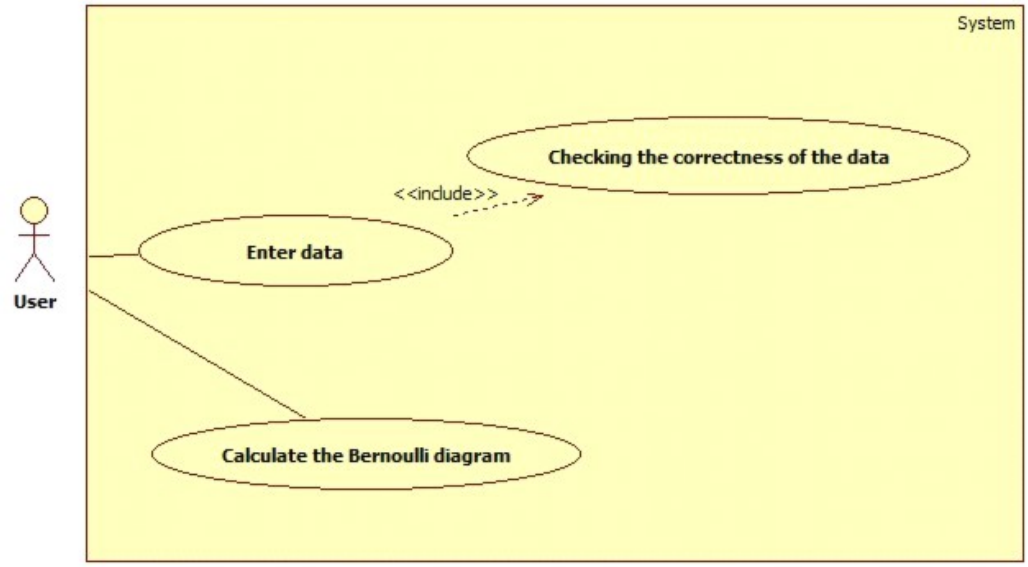
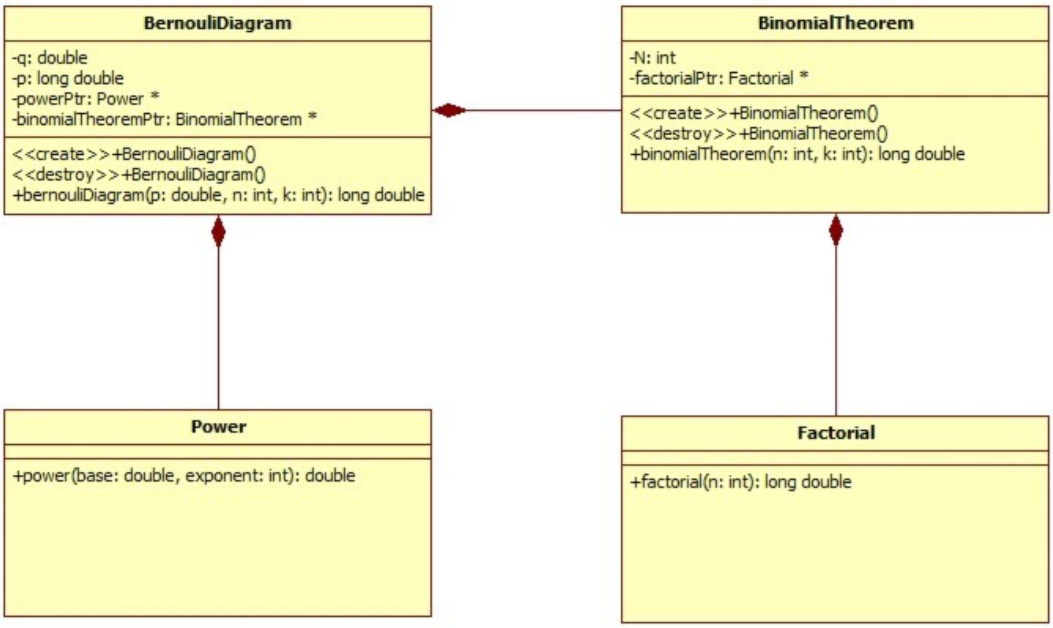
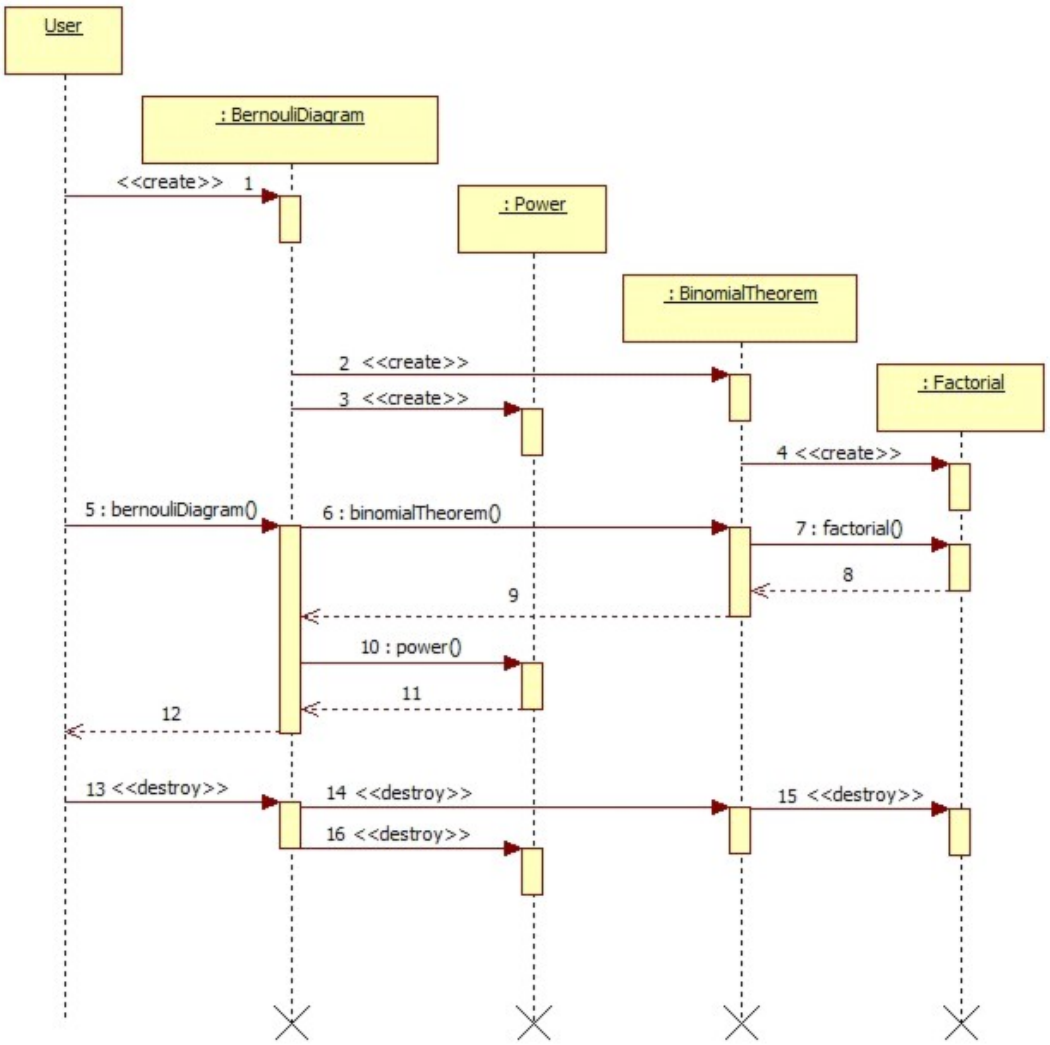

System
Checking the correctness of the data
<<include>>
Enter data
User
Calculate the Bernoulli diagram


BernouliDiagram
-q: double
-p: long double
-powerPtr: Power *
-binomialTheoremPtr: BinomialTheorem *
<<create>>+BernouliDiagram()
<<destroy>>+BernouliDiagram()
+bernouliDiagram(p: double, n: int, k: int): long double
BinomialTheorem
-N: int
-factorialPtr: Factorial *
<<create>>+BinomialTheorem()
<<destroy>>+BinomialTheorem()
+binomialTheorem(n: int, k: int): long double
Power
+power(base: double, exponent: int): double
Factorial
+factorial(n: int): long double




```
//
//
// Generated by StarUML(tm) C++ Add-In
//
// @ Project : Laboratorium 07
// @ File Name : BernouliDiagram.h
// @ Date : 30.05.2023
// @ Author : Tomasz Wnuk
//
//


#if !defined(_BERNOULIDIAGRAM_H)
```

```
#define _BERNOULIDIAGRAM_H


#include "BinomialTheorem.h"
#include "Power.h"


class BernouliDiagram {
public:
   long double bernouliDiagram(double p, int n, int k);
   BernouliDiagram();
   ~BernouliDiagram();
private:
                double q;
                long double p;
                Power * powerPtr;
                BinomialTheorem * binomialTheoremPtr;
};


#endif  //_BERNOULIDIAGRAM_H
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 07
//  @ File Name : BernouliDiagram.cpp
//  @ Date : 30.05.2023
//  @ Author : Tomasz Wnuk
//
//
```

```
#include "BernouliDiagram.h"


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


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


long double BernouliDiagram::bernouliDiagram(double p, int n, int k) {
   q = 1 - p;
   return ((binomialTheoremPtr->binomialTheorem(n, k)) * (powerPtr->power(p, k)) * powerPtr->power(q,
n - k));
}
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 07
//  @ File Name : BinomialTheorem.h
//  @ Date : 30.05.2023
//  @ Author : Tomasz Wnuk
//
//


#if !defined(_BINOMIALTHEOREM_H)
```

```
#define _BINOMIALTHEOREM_H

#include "Factorial.h"

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

#endif  //_BINOMIALTHEOREM_H
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 07
//  @ File Name : BinomialTheorem.cpp
//  @ Date : 30.05.2023
//  @ Author : Tomasz Wnuk
//
//

#include "BinomialTheorem.h"

BinomialTheorem::BinomialTheorem() {
```

```
    factorialPtr = new Factorial();
}

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

long double BinomialTheorem::binomialTheorem(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->factorial(n - k));
    } else {
        for(int i = N + 1; i <= n; i++) {
            NbyK *= i;
        }
        return (NbyK / factorialPtr->factorial(k));
    }
}
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 07
//  @ File Name : Factorial.h
```

```
//  @ Date : 30.05.2023
//  @ Author : Tomasz Wnuk
//
//


#if !defined(_FACTORIAL_H)
#define _FACTORIAL_H

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

#endif  //_FACTORIAL_H
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 07
//  @ File Name : Factorial.cpp
//  @ Date : 30.05.2023
//  @ Author : Tomasz Wnuk
//
//


#include "Factorial.h"

long double Factorial::factorial(int n) {
   if (n == 0) {
```

```
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}
//
//
//  Generated by StarUML(tm) C++ Add-In
//
//  @ Project : Laboratorium 07
//  @ File Name : Power.h
//  @ Date : 30.05.2023
//  @ Author : Tomasz Wnuk
//
//

#if !defined(_POWER_H)
#define _POWER_H

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

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

```
//  @ Project : Laboratorium 07
//  @ File Name : Power.cpp
//  @ Date : 30.05.2023
//  @ Author : Tomasz Wnuk
//
//

#include "Power.h"

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


// Print result

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


// Get user input

string userInput;

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;
}
```

```
--------------------------------------------------------------------------------
----------------------------Calculating Bernouli Diagram----------------------------
--------------------------------------------------------------------------------
Enter probability of success in a single experiment [p]: 0.5
Enter number of experiments in the Bernoulli diagram [n]: 6
Enter number of experiments ending with success in the Bernoulli diagram [k]: 2
Probability of 2 successes in 6 experiments with probability of success in a single experiment equal to 0.5 is equal to 0.234375
Do you want to continue? (y/n): n

Process finished with exit code 0
```

wyk. Tomasz Wnuk