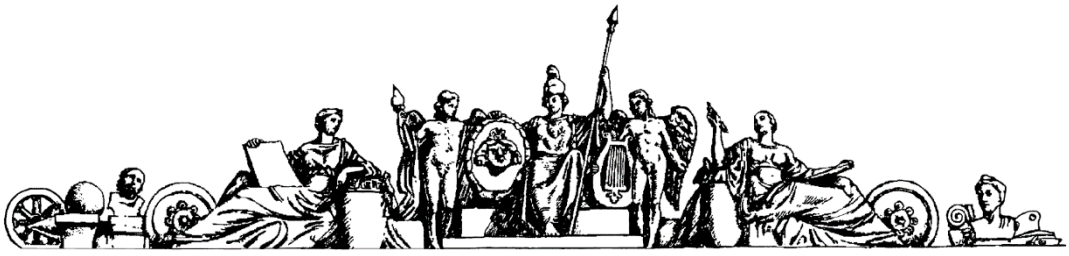


Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)
Факультет «Робототехника и комплексная автоматизация» (РК)
Кафедра «Системы автоматизированного проектирования» (РК6)

Домашнее задание №2 (часть 1) по «Теории вероятности».

Студент: Сергеева Диана

Группа: РК6-36Б

Преподаватель: Берчун Ю.В

Проверил:

Дата:

2020

Задача 1. Рассматривается извлечение шаров с возвращением из первой корзины (см. исходные данные к ДЗ №1: R1=8, G1=7, B1=5 – 18 вариант). Выполняется серия из n экспериментов, подсчитывается число k извлечений красных шаров.

1. Построить графики вероятности P(k). Графики строятся для числа опытов n = 6, 9 и 12 с расчётом вероятностей по формуле Бернулли.
2. Для n = 6 также строится график функции распределения F(x).
3. Для n = 25, 50, 100, 200, 400, 1000 строится огибающая графика P(k), при этом для каждого графика рассчитываются не менее 7 точек с использованием локальной теоремы Муавра-Лапласа.
4. Построить график вероятности того, что абсолютное число извлечений красных шаров отклонится от математического ожидания не более, чем на R1. При построении графика использовать n = 25, 50, 100.
5. Построить график вероятности того, что относительное число извлечений красных шаров отклонится от математического ожидания не более, чем на R1 / (R1+G1+B1). При построении графика использовать n = 100, 200, 400.
6. Рассчитать допустимый интервал числа успешных испытаний k (симметричный относительно математического ожидания), обеспечивающий попадание в него с вероятностью P = R1 / (R1+G1+B1) при n = 1000.
7. Построить график зависимости минимально необходимого числа испытаний n для того, чтобы обеспечить вероятность появления не менее, чем N1=R1+G1+B1 красных шаров с вероятностями P = 0,7; 0,8; 0,9; 0,95.

1.1 Формула Бернулли: $P_n(k) = C_n^k * p^k * q^{n-k}$

Для n=6:

```
P(0)= 0.046656
P(1)= 0.186624
P(2)= 0.31104
P(3)= 0.27648
P(4)= 0.13824
P(5)= 0.036864
P(6)= 0.004096
```

Для n=9 :

```
P(0)= 0.0100777
P(1)= 0.0604662
P(2)= 0.161243
P(3)= 0.250823
P(4)= 0.250823
P(5)= 0.167215
P(6)= 0.0743178
P(7)= 0.0212337
P(8)= 0.00353894
P(9)= 0.000262144
```

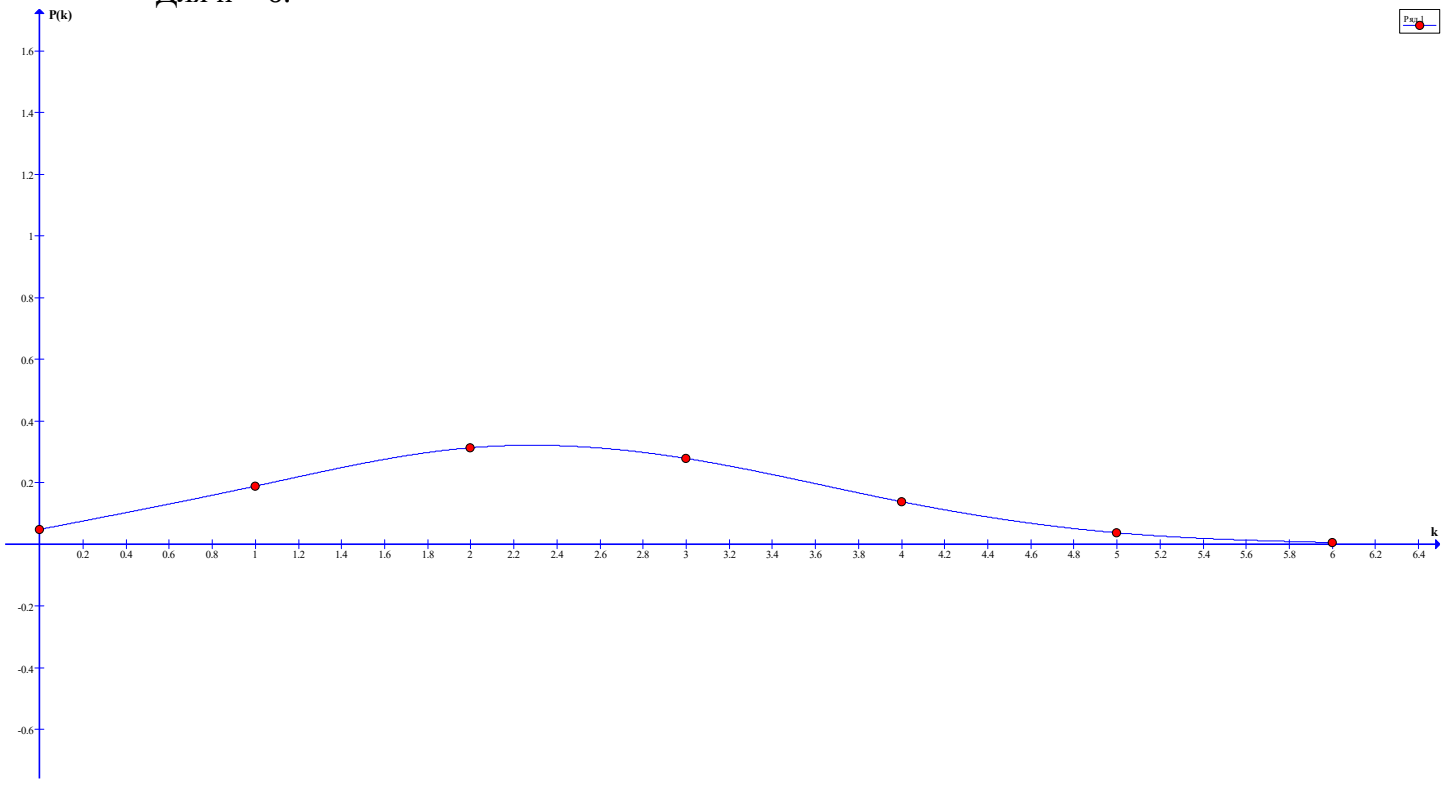
Для n = 12:

```
P(0)= 0.00217678
P(1)= 0.0174143
P(2)= 0.0638523
P(3)= 0.141894
P(4)= 0.212841
P(5)= 0.22703
P(6)= 0.176579
P(7)= 0.100902
P(8)= 0.0420427
P(9)= 0.0124571
P(10)= 0.00249142
P(11)= 0.00030199
P(12)= 1.67772e-05
```

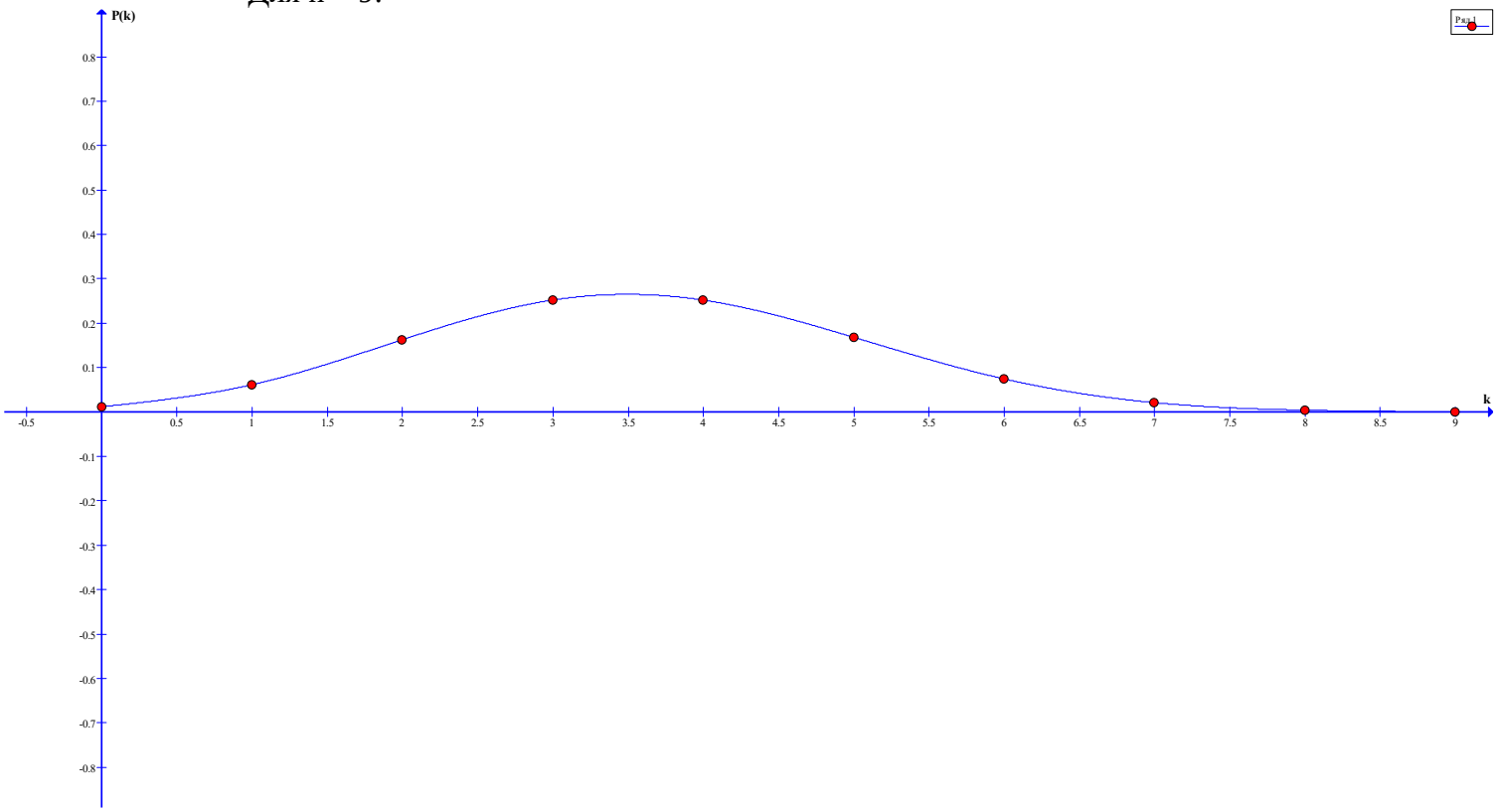
Программа для вычислений:

```cpp
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

int factorial(int a)
{
    int res = 1;
    for (int i = a; i > 1; --i)
    {
        res *= i;
    }
    return res;
}

double comb(int n, int k)
{
    return (double)factorial(n) / ((double)factorial(k) * (double)factorial(n - k));
}

int main()
{
    int n;
    cin >> n;

    int r1 = 8, g1 = 7, b1 = 5;
    double p = (double)r1 / (double)(r1 + g1 + b1);

    cout.precision(2);
    for (int i = 0; i <= n; ++i)
    {
        double P = comb(n, i) * pow(p, i) * pow((1 - p), (n - i));
        cout << "P("  <<  i << ")= " << setprecision(6) <<  P << endl;
    }
    return 0;
}
```

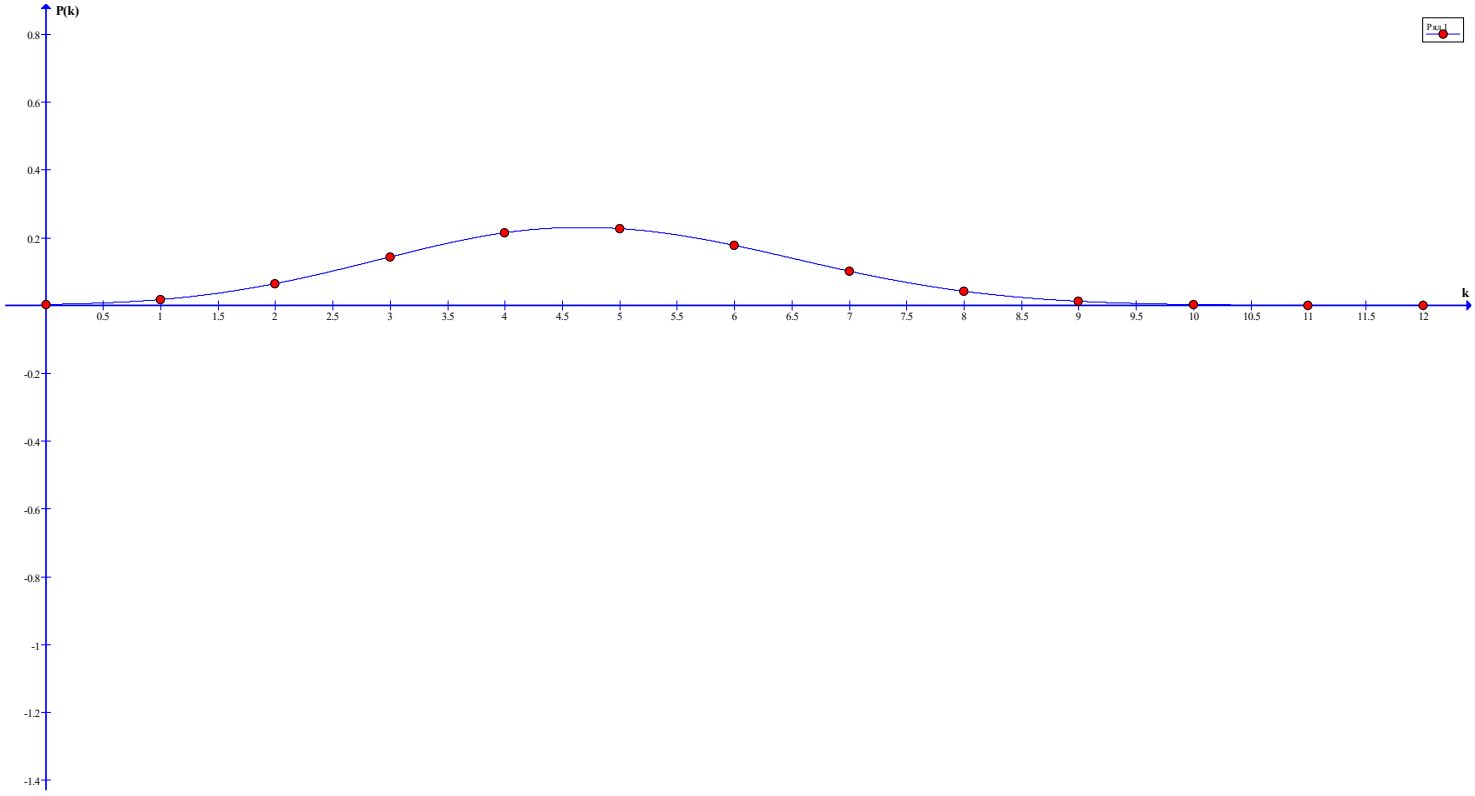
Для n = 6:



Для n = 9:



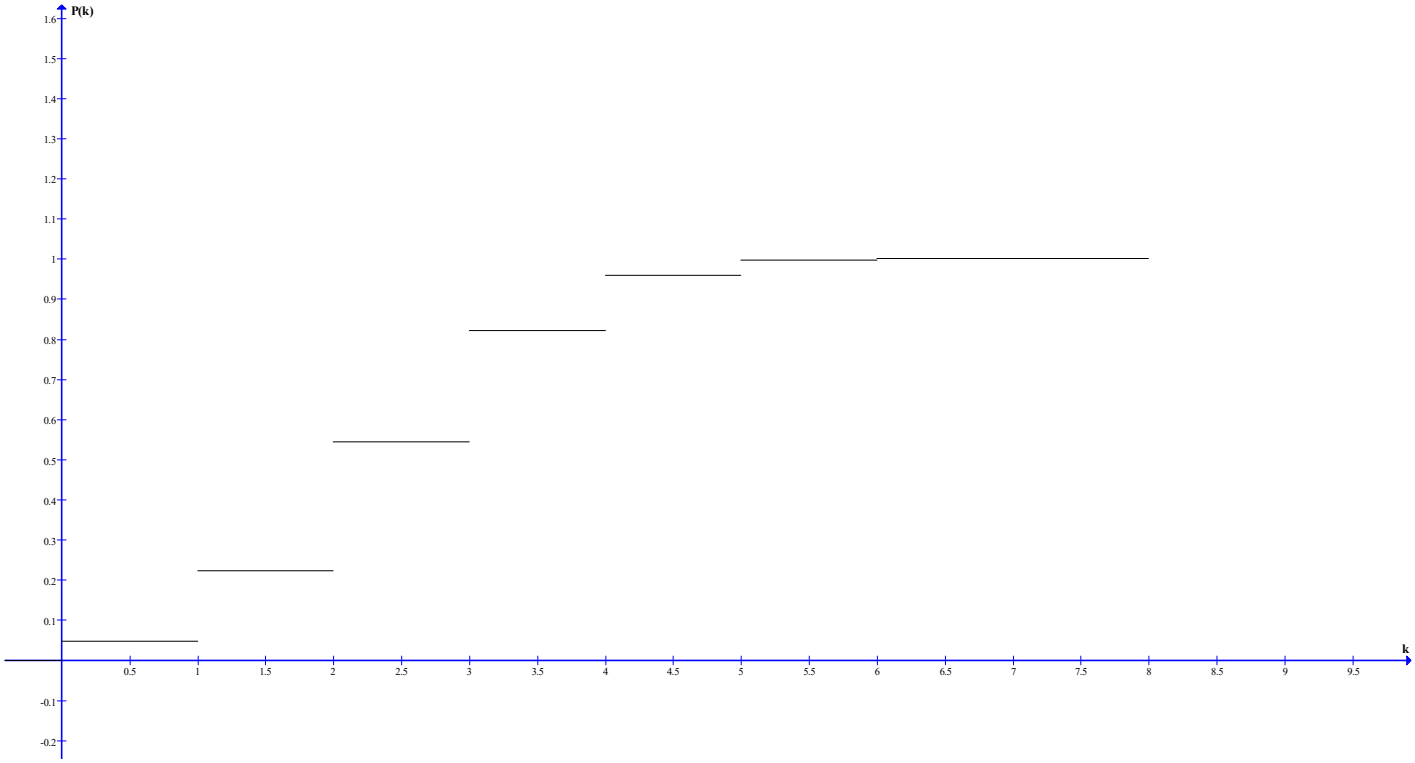
Для n = 12:



1.2 Для n=6:

```
P(0)= 0.046656
P(1)= 0.186624
P(2)= 0.31104
P(3)= 0.27648
P(4)= 0.13824
P(5)= 0.036864
P(6)= 0.004096
```

Для $x \leq 0$: F(x) = 0;
Для $0 < x \leq 1$: F(x) = 0.046656;
Для $1 < x \leq 2$: F(x) = 0.046656 + 0.186624 = 0.23328;
Для $2 < x \leq 3$: F(x) = 0.23328 + 0.31104 = 0.54432;
Для $3 < x \leq 4$: F(x) = 0.54432 + 0.27648 = 0.8202;
Для $4 < x \leq 5$: F(x) = 0.8202 + 0.13824 = 0.95904;
Для $5 < x \leq 6$: F(x) = 0.95904 + 0.036864 = 0.995904;
Для $6 < x$: F(x) = 0.995904 + 0.004096 = 1



1.3 Теорема Муавра-Лапласа: $P_n(k) = \frac{1}{\sqrt{npq}} \phi\left(\frac{k-np}{\sqrt{npq}}\right)$, $\phi(x) = \frac{1}{\sqrt{2\pi}} * e^{\frac{-x^2}{2}}$

```
n = 25
F(-4.082) = 0.0001
P(0) = 4.082e-05

F(-2.858) = 0.0067
P(3) = 0.002735

F(-1.633) = 0.1057
P(6) = 0.04315

F(-0.4082) = 0.3668
P(9) = 0.1497

F(0.8165) = 0.2850
P(12) = 0.1164

F(2.041) = 0.0498
P(15) = 0.02033

F(3.266) = 0.0019
P(18) = 0.0007757
```

```
n = 50

F(-5.774) = 0.0001
P(0) = 2.887e-05

F(-3.753) = 0.0004
P(7) = 0.0001155

F(-1.732) = 0.0893
P(14) = 0.02578

F(0.2887) = 0.3825
P(21) = 0.1104

F(2.309) = 0.0277
P(28) = 0.007996

F(4.33) = 0.0001
P(35) = 2.887e-05

F(6.351) = 0.0001
P(42) = 2.887e-05
```

```
n = 100

F(-8.165) = 0.0001
P(0) = 2.041e-05

F(-5.307) = 0.0001
P(14) = 2.041e-05

F(-2.449) = 0.0198
P(28) = 0.004042

F(0.4082) = 0.3668
P(42) = 0.07487

F(3.266) = 0.0019
P(56) = 0.0003878

F(6.124) = 0.0001
P(70) = 2.041e-05

F(8.981) = 0.0001
P(84) = 2.041e-05
```

```
n = 200

F(-11.55) = 0.0001
P(0) = 1.443e-05

F(-7.506) = 0.0001
P(28) = 1.443e-05

F(-3.464) = 0.0010
P(56) = 0.0001443

F(0.5774) = 0.3372
P(84) = 0.04867

F(4.619) = 0.0001
P(112) = 1.443e-05

F(8.66) = 0.0001
P(140) = 1.443e-05

F(12.7) = 0.0001
P(168) = 1.443e-05
```

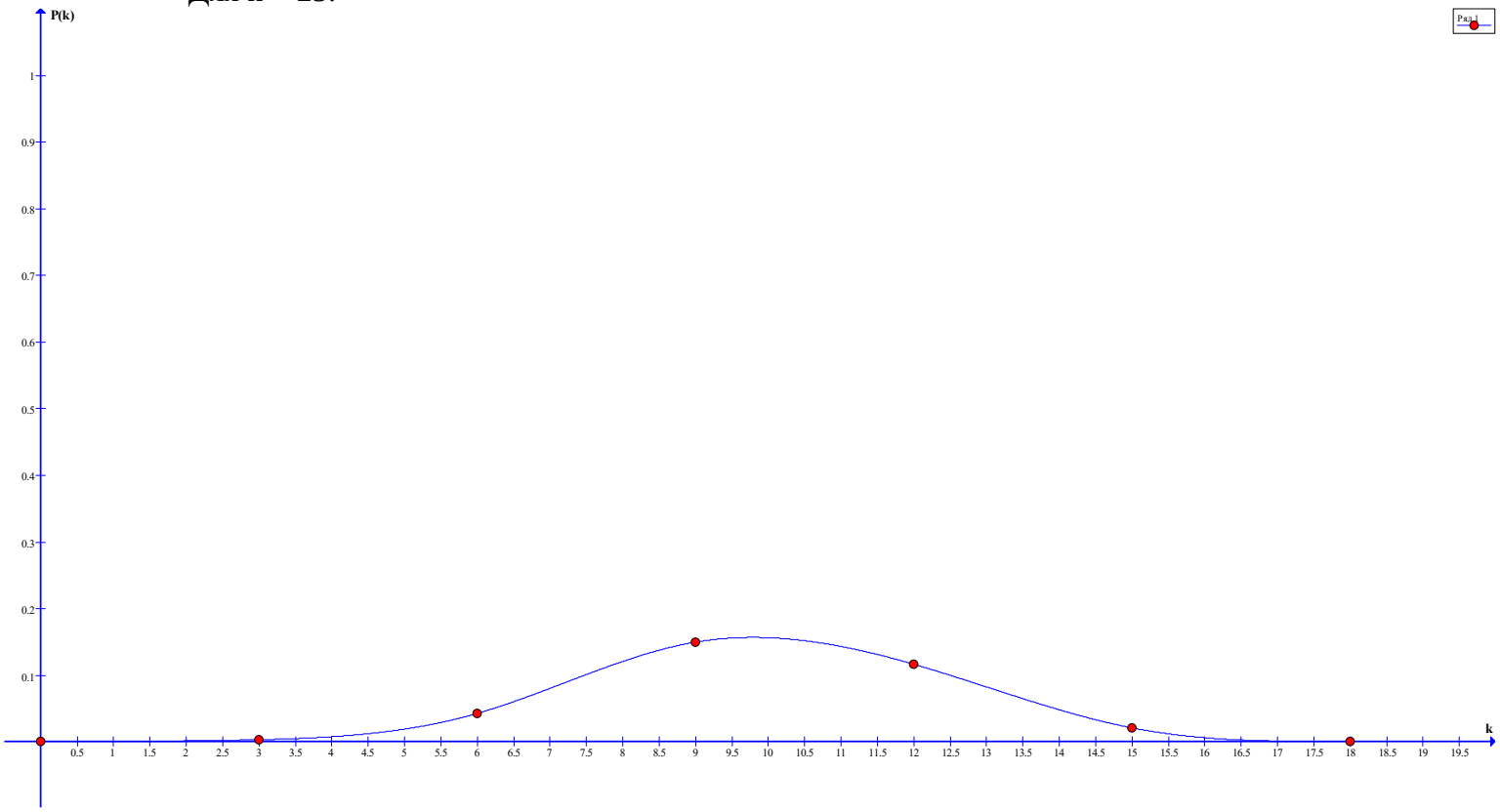
```
n = 400

F(-16.33) = 0.0001
P(0) = 1.021e-05

F(-10.51) = 0.0001
P(57) = 1.021e-05

F(-4.695) = 0.0001
P(114) = 1.021e-05

F(1.123) = 0.2131
P(171) = 0.02175

F(6.94) = 0.0001
P(228) = 1.021e-05

F(12.76) = 0.0001
P(285) = 1.021e-05

F(18.58) = 0.0001
P(342) = 1.021e-05
```

```
n = 1000

F(-25.82) = 0.0001
P(0) = 6.455e-06

F(-16.65) = 0.0001
P(142) = 6.455e-06

F(-7.488) = 0.0001
P(284) = 6.455e-06

F(1.678) = 0.0973
P(426) = 0.006281

F(10.84) = 0.0001
P(568) = 6.455e-06

F(20.01) = 0.0001
P(710) = 6.455e-06

F(29.18) = 0.0001
P(852) = 6.455e-06
```

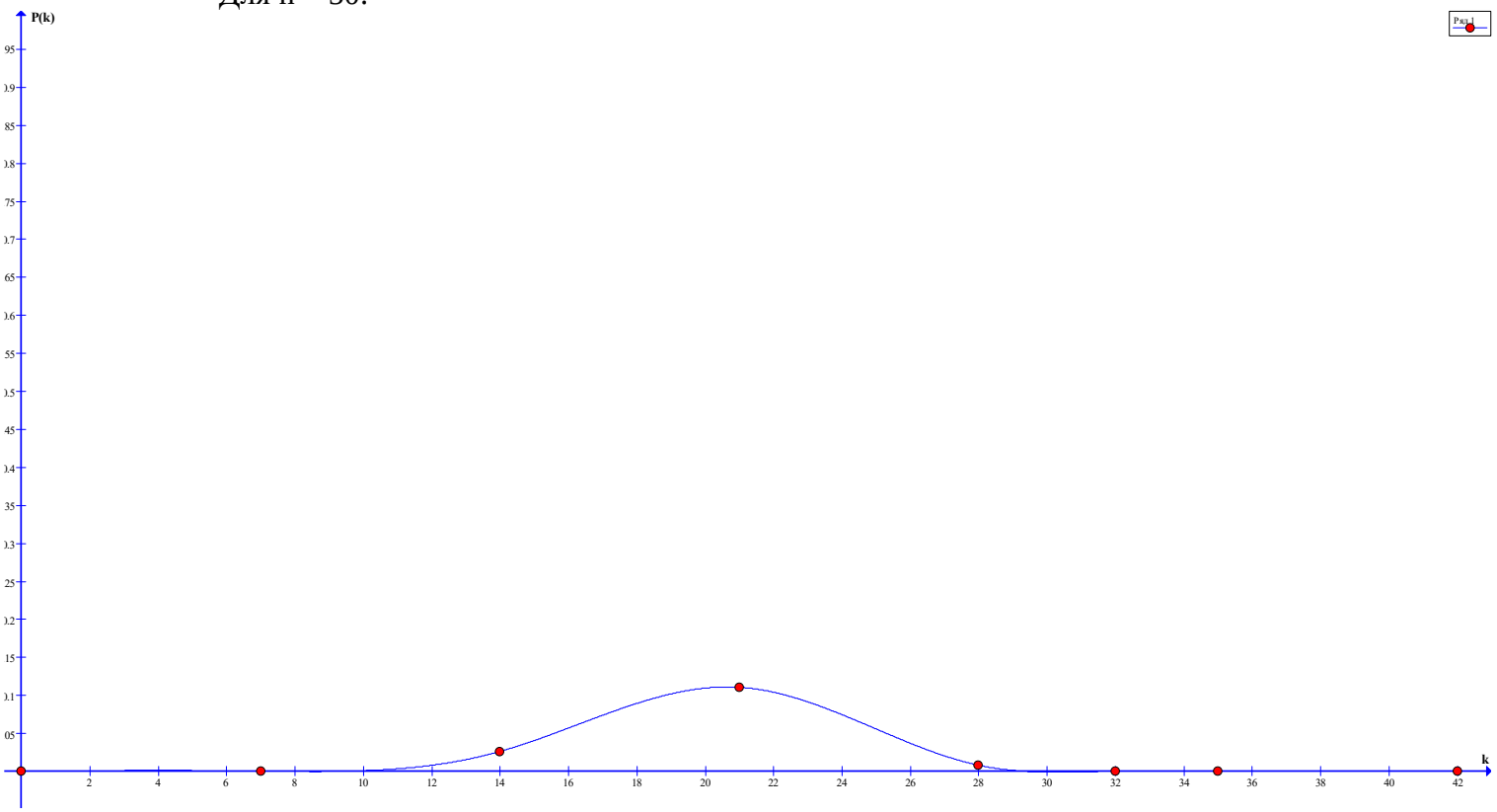
Программа для вычислений:

```cpp
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;

int main()
{
    int n;
    cout << "n = ";
    cin >> n;
    cout << endl;

    int r = 8, g = 7, b = 5;
    double p = (double)r / (double)(r + g + b);
    double q = 1 - p;

    int addition = n / 7, summ = 0;
    for (int i = 0; i < 7; ++i)
    {
        double x = ((double)summ - (double)n * p) / sqrt((double)n * p * q);
        cout << "F(" << setprecision(4) << x << ") = ";

        double tabularValue;
        cin >> tabularValue;

        double P;
        P = tabularValue / sqrt((double)n * p * q);
        cout << "P(" << summ << ") = " << setprecision(4) << P << endl;
        cout << endl;

        summ += addition;
    }

    return 0;
}
```

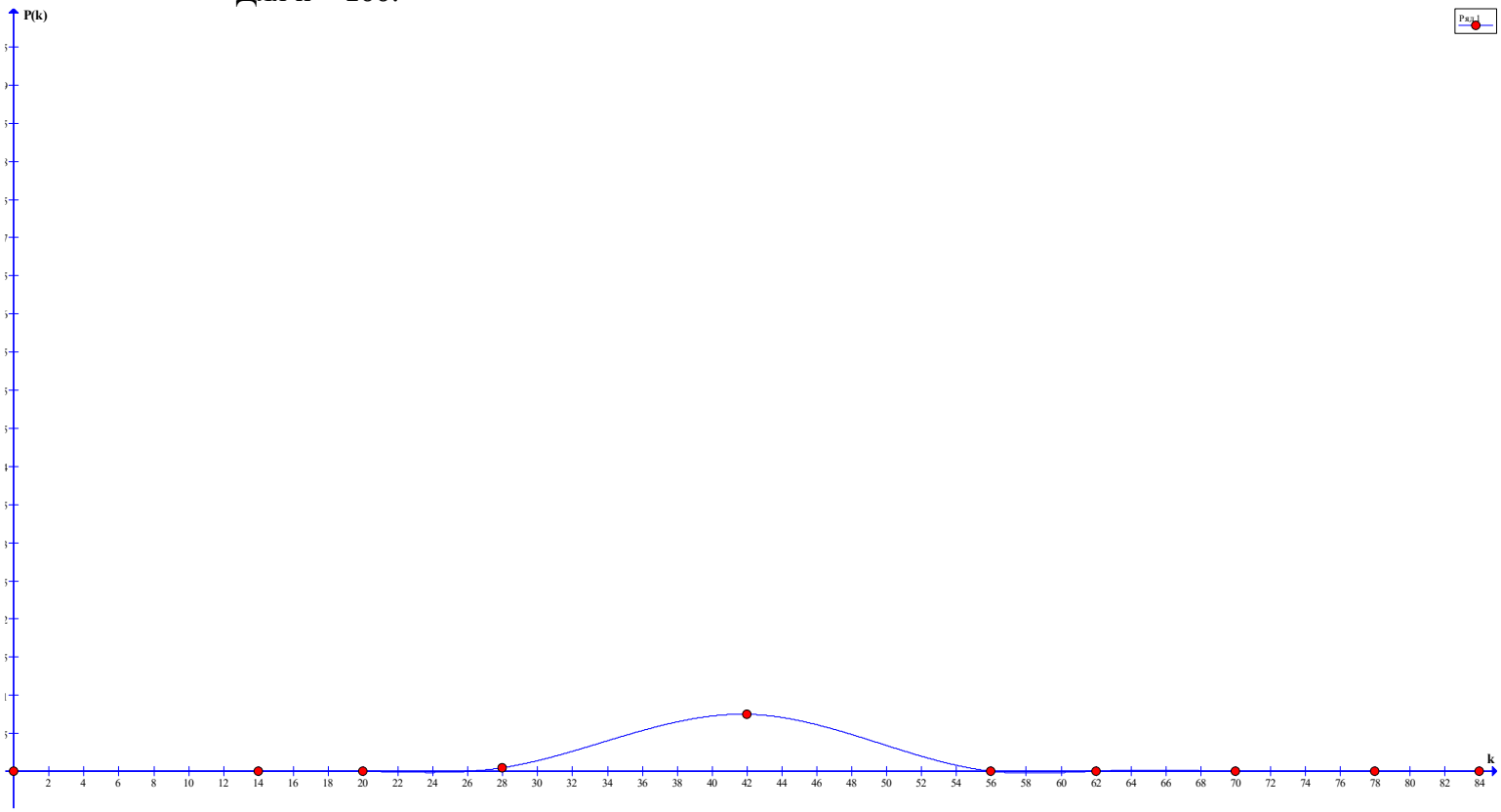
Для n = 25:



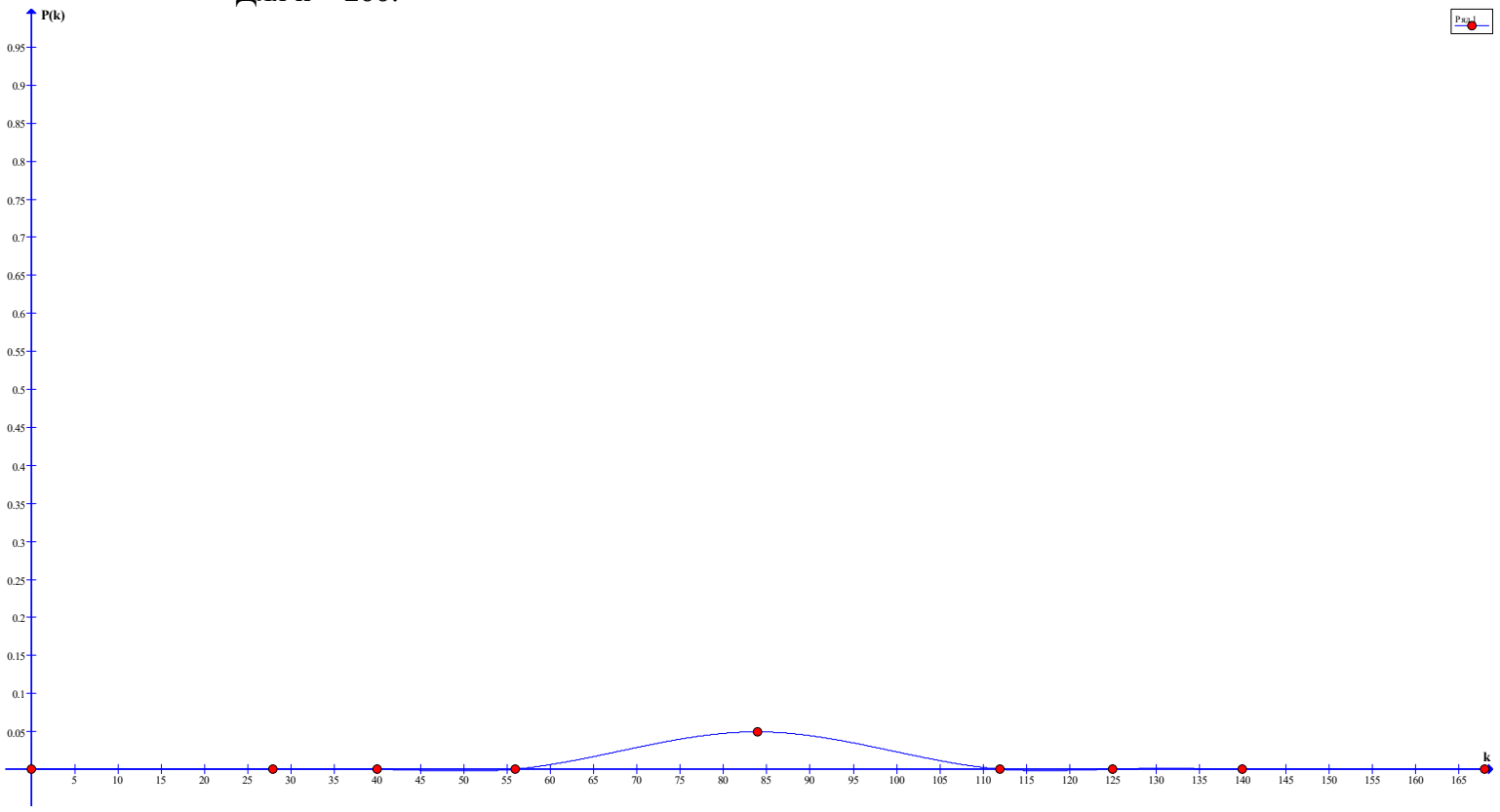
Для n = 50:



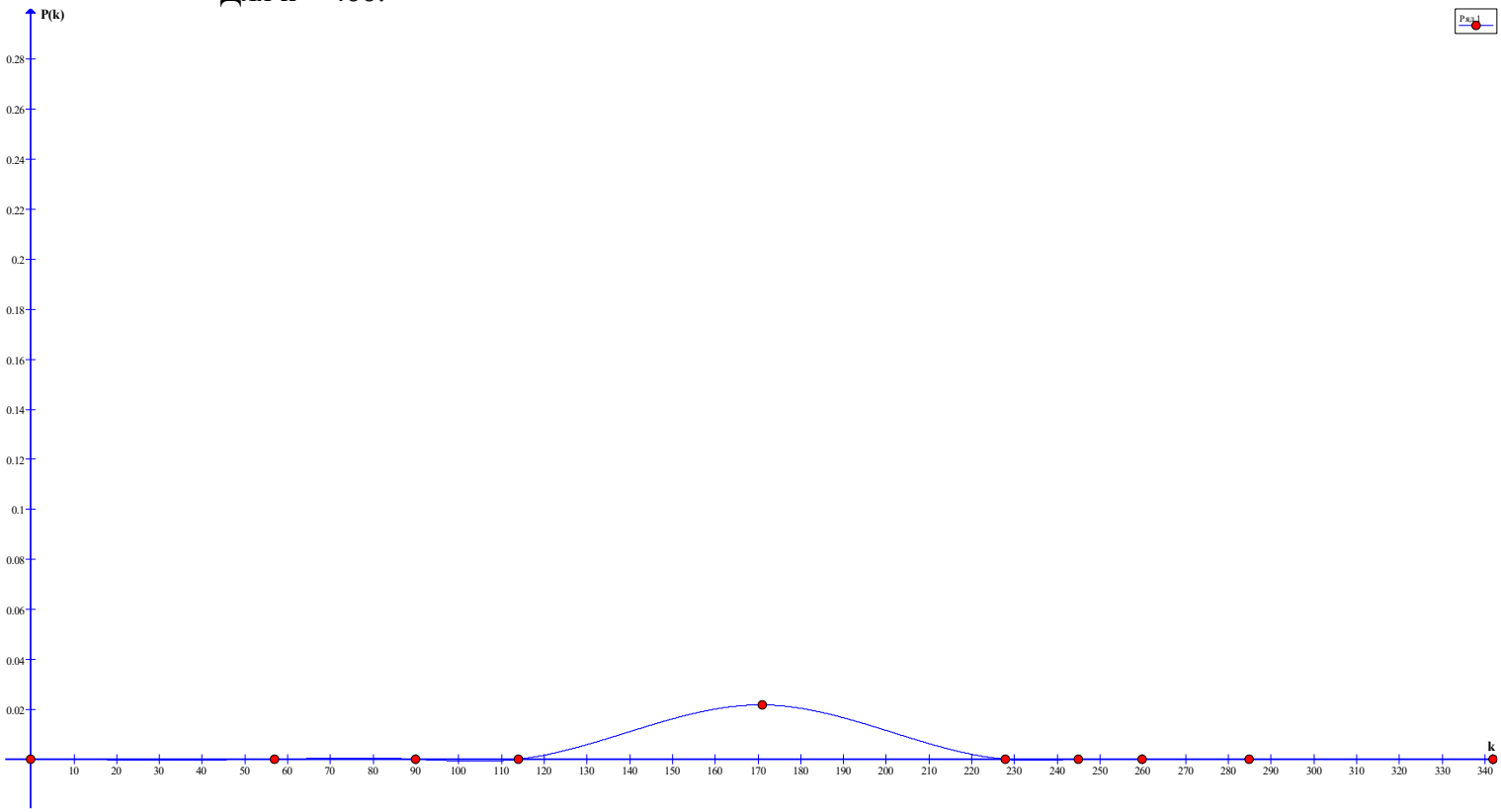
Для n = 100:



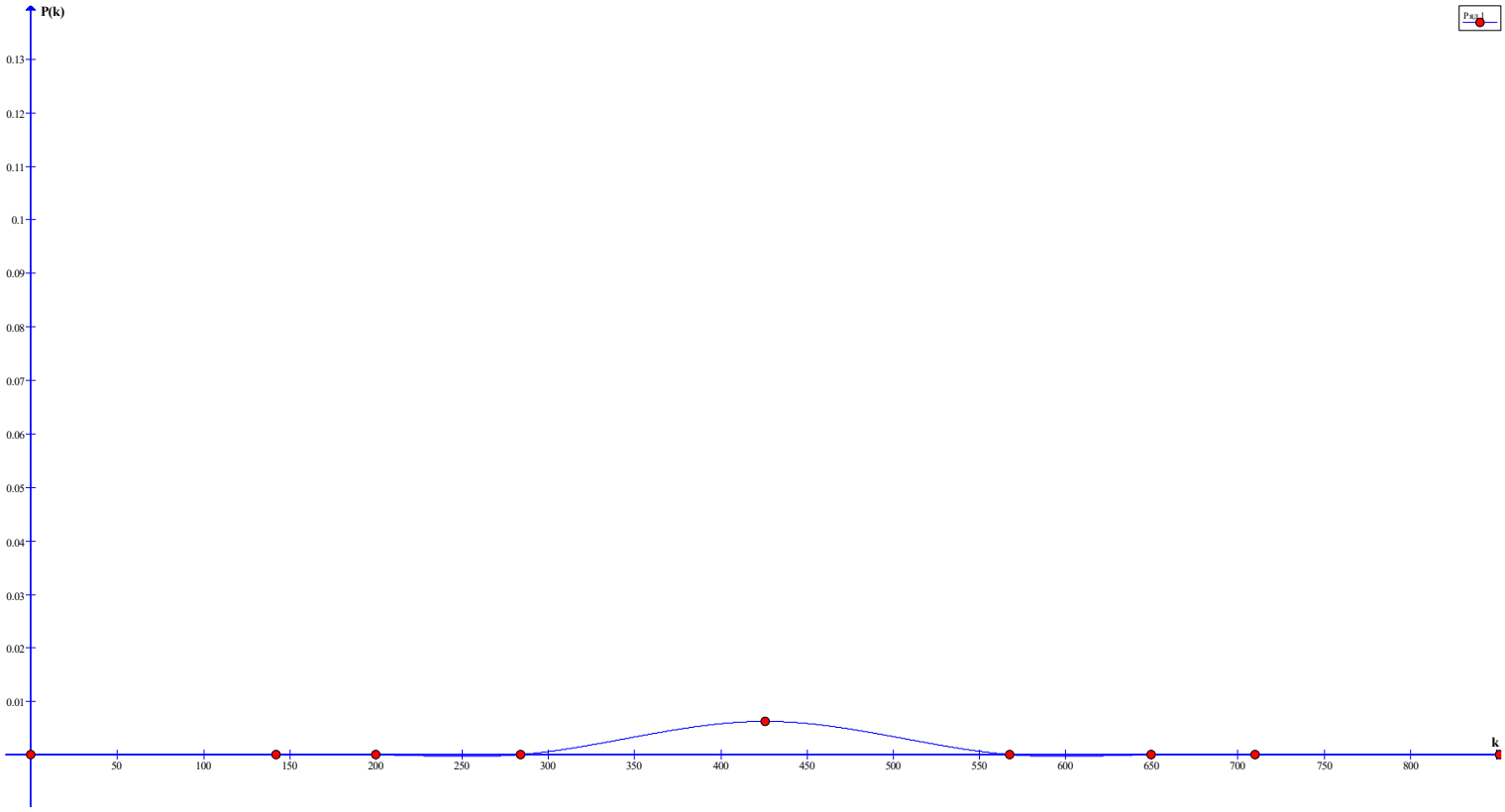
Для n = 200:



Для n = 400:



Для n = 1000:



1.4 $P(|k - M| < R1) = 2F(\frac{R1}{\sigma}) = 2F(\frac{R1}{\sqrt{D(k)}}) = 2F(\frac{R1}{\sqrt{(M(k^2)-(M(k))^2)}})$

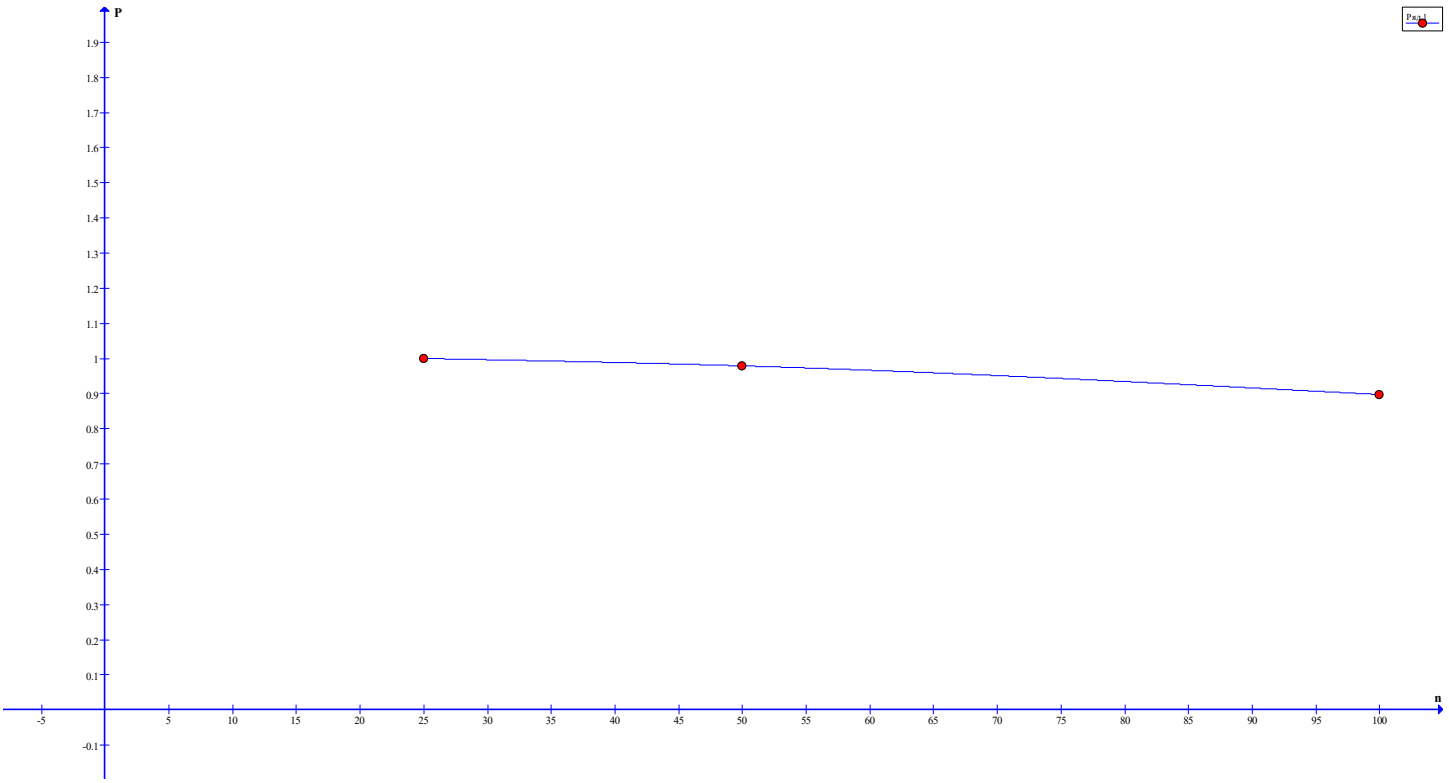
```
n = 25

F(3.266) = 0.49931
P(|k - M| < R1) = 0.9986
```

```
n = 50

F(2.309) = 0.4893
P(|k - M| < R1) = 0.9786
```

```
n = 100

F(1.633) = 0.4484
P(|k - M| < R1) = 0.8968
```

Программа для вычислений:

```cpp
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;

double comb(int n, int k)
{
    if ((k == 0) || (k == n))
        return 1;
    if ((k == 1) || (k == n - 1))
        return n;

    int numStart;
    if (k > (n / 2))
        numStart = k + 1;
    else
        numStart = (n - k) + 1;

    int denom;
    if (k > (n / 2))
        denom = n - k;
    else
        denom = k;

    int amount;
    if (denom > (n - numStart))
        amount = n - numStart + 1;
    else
        amount = denom;

    double res = 1;
    int divider = 1, multiplier = numStart;
    for (int i = 1; i <= amount; ++i)
    {
        res *= multiplier;
        res /= divider;

        multiplier++;
        divider++;
    }

    return res;
}

double functionBernulli(int n, int k, double p, double q)
{
    return comb(n, k) * pow(p, k) * pow(q, n - k);
}

double dispersion(int n, double p, double q)
{
    double res1 = 0, res2 = 0;
    for (int k = 0; k <= n; ++k)
    {
        double Bernulli = functionBernulli(n, k, p, q);
        res1 += (double)(k * k) * Bernulli;
        res2 += (double)(k) * Bernulli;
    }

    res2 = pow(res2, 2);
    return (res1 - res2);
}

int main()
{
    int n;
    cout << "n = ";
    cin >> n;
    cout << endl;

    int r = 8, g = 7, b = 5;
    double p = (double)r / (double)(r + g + b);
    double q = 1 - p;

    double x = (double)r / sqrt(dispersion(n, p, q));
    cout << "F(" << setprecision(4) << x << ") = ";
    double F;
    cin >> F;

    cout << "P(|k - M| < R1) = " << setprecision(4) << 2 * F << endl;

    return 0;
}
```

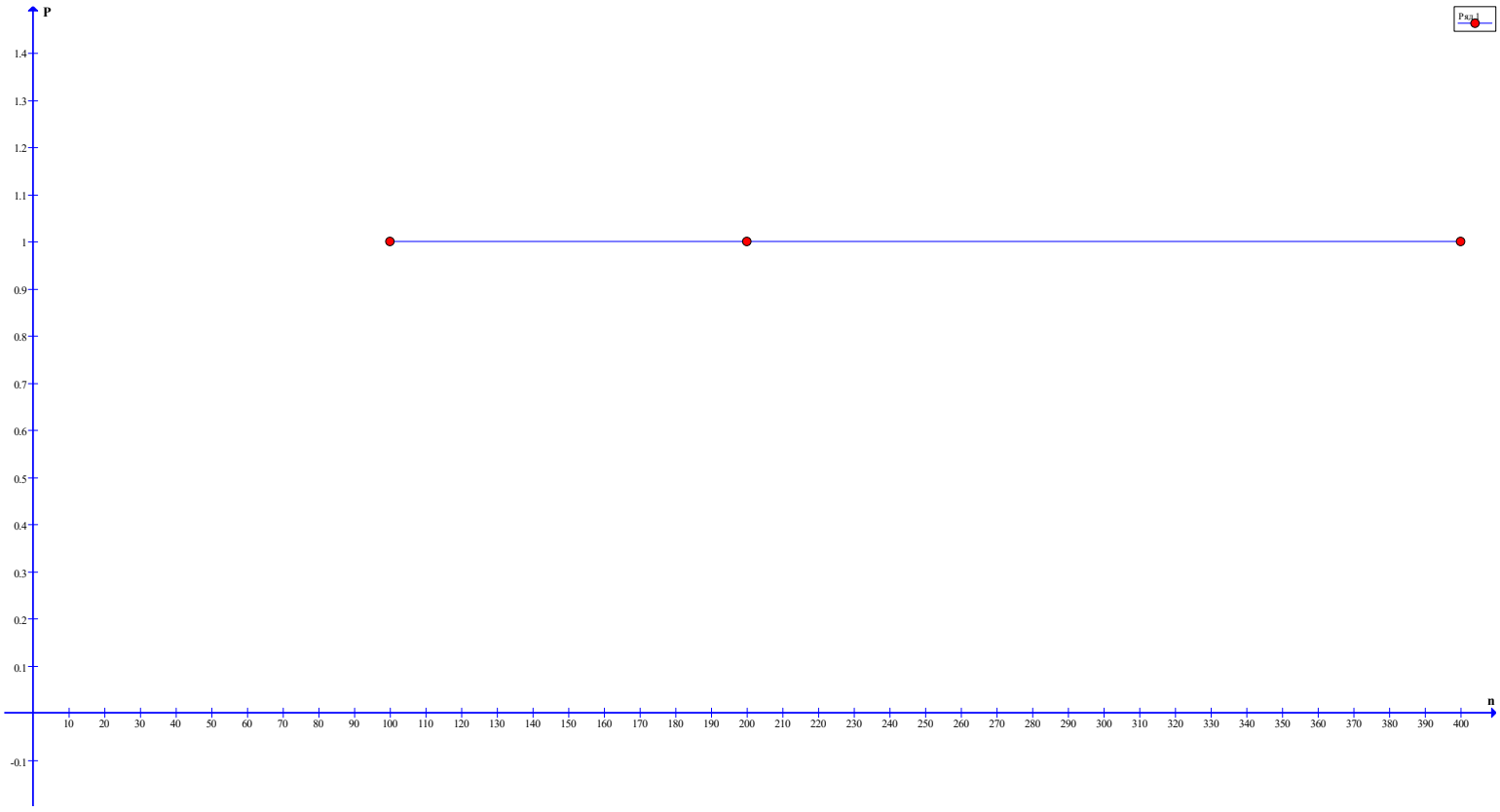


1.5 $P = 2F\left(\frac{\frac{R1}{R1+G1+B1}}{\sigma}\right) = 2F\left(\frac{\frac{R1}{R1+G1+B1}}{\sqrt{D(k)}}\right) = 2F\left(\frac{\frac{R1}{R1+G1+B1}}{\sqrt{(M(k^2)-(M(k))^2)}}\right)$

```
n = 100

F(8.165) = 0.49999
P = 0.9999
```

```
n = 200

F(11.55) = 0.49999
P = 0.9999
```

```
n = 400

F(16.33) = 0.4999
P = 0.9999
```



Программа для вычислений:

```cpp
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;

double comb(int n, int k)
{
    if ((k == 0) || (k == n))
        return 1;
    if ((k == 1) || (k == n - 1))
        return n;

    int numStart;
    if (k > (n / 2))
        numStart = k + 1;
    else
        numStart = (n - k) + 1;

    int denom;
    if (k > (n / 2))
        denom = n - k;
    else
        denom = k;

    int amount;
    if (denom > (n - numStart))
        amount = n - numStart + 1;
    else
        amount = denom;

    double res = 1;
    int divider = 1, multiplier = numStart;
    for (int i = 1; i <= amount; ++i)
    {
        res *= multiplier;
        res /= divider;

        multiplier++;
        divider++;
    }

    return res;
}

double functionBernulli(int n, int k, double p, double q)
{
    return comb(n, k) * pow(p, k) * pow(q, n - k);
}

double dispersionRelative(int n, double p, double q)
{
    double res1 = 0, res2 = 0;
    for (int k = 0; k <= n; ++k)
    {
        double Bernulli = functionBernulli(n, k, p, q);
        res1 += Bernulli * (double)(k * k) / (double)(n * n);
        res2 += Bernulli * (double)(k) / (double)n;
    }

    res2 = pow(res2, 2);
    return (res1 - res2);
}

int main()
{
    int n;
    cout << "n = ";
    cin >> n;
    cout << endl;

    int r = 8, g = 7, b = 5;
    double p = (double)r / ((double)r + (double)g + (double)b);
    double q = 1 - p;

    double x = p / sqrt(dispersionRelative(n, p, q));
    cout << "F(" << setprecision(4) << x << ") = ";
    double F;
    cin >> F;

    cout << "P = " << setprecision(4) << 2 * F << endl;

    return 0;
}
```

$$1.6\ P = 2 * \varphi\left(\frac{\delta}{\sigma}\right);\ \varphi\left(\frac{\delta}{\sigma}\right) = \frac{P}{2} = \frac{R1}{(R1+G1+B1)*2} = \frac{8}{20*2} = 0.2 \Rightarrow \varphi\left(\frac{\delta}{\sigma}\right) = 0.2$$

$$\frac{\delta}{\sigma} = 0.05;\ \delta = 0.05 * \sigma, \text{где } \sigma = 15.49;\ \delta = 0.7745;\ 399 < k < 401$$

```cpp
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;

double comb(int n, int k)
{
    if ((k == 0) || (k == n))
        return 1;
    if ((k == 1) || (k == n - 1))
        return n;

    int numStart;
    if (k > (n / 2))
        numStart = k + 1;
    else
        numStart = (n - k) + 1;

    int denom;
    if (k > (n / 2))
        denom = n - k;
    else
        denom = k;

    int amount;
    if (denom > (n - numStart))
        amount = n - numStart + 1;
    else
        amount = denom;

    double res = 1;
    int divider = 1, multiplier = numStart;
    for (int i = 1; i <= amount; ++i)
    {
        res *= multiplier;
        res /= divider;

        multiplier++;
        divider++;
    }

    return res;
}

double functionBernulli(int n, int k, double p, double q)
{
    return comb(n, k) * pow(p, k) * pow(q, n - k);
}

double dispersion(int n, double p, double q)
{
    double res1 = 0, res2 = 0;
    for (int k = 0; k <= n; ++k)
    {
        double Bernulli = functionBernulli(n, k, p, q);
        res1 += (double)(k * k) * Bernulli;
        res2 += (double)(k) * Bernulli;
    }

    cout << "Mathematical Expected: " << setprecision(4) << res2 << endl;

    res2 = pow(res2, 2);
    return (res1 - res2);
}
```

```cpp
int main()
{
    int n;
    cout << "n = ";
    cin >> n;
    cout << endl;

    int r = 8, g = 7, b = 5;
    double p = (double)r / ((double)r + (double)g + (double)b);
    double q = 1 - p;

    double sigma =  sqrt(dispersion(n, p, q));
    cout << "sigma = " << setprecision(4) << sigma << endl;


    return 0;
}
```

1.7 $N1 = R1 + G1 + B1 = 8 + 7 + 5 = 20$

$$P(N1 \le k \le n) = F\left(\frac{n-np}{\sqrt{npq}}\right) - F\left(\frac{N1-np}{\sqrt{npq}}\right) = F\left(\frac{n-n\frac{8}{20}}{\sqrt{n*\frac{8}{20}*\frac{12}{20}}}\right) - F\left(\frac{20-n\frac{8}{20}}{\sqrt{n*\frac{8}{20}*\frac{12}{20}}}\right) =$$

$$F\left(\frac{\sqrt{0.6*0.6n^2}}{\sqrt{n*0.4*0.6}}\right) - F\left(\frac{400-8n}{20\sqrt{n*0.24}}\right) = F\left(\sqrt{1.5n}\right) - F\left(\frac{400-8n}{20\sqrt{n*0.24}}\right)$$

При n ≥17: $\left(\sqrt{1.5n}\right) = 5 \Rightarrow F\left(\sqrt{1.5n}\right) = 0.5$

$$P(N1 \le k \le n) = 0.5 - F\left(\frac{400-8n}{20\sqrt{n*0.24}}\right)$$

P = 0.7:

$$-0.2 = F\left(\frac{400-8n}{20\sqrt{n*0.24}}\right) \Rightarrow \frac{400-8n}{20\sqrt{n*0.24}} = -0.53 \Rightarrow \text{n} = 56$$

P = 0.8:

$$-0.3 = F\left(\frac{400-8n}{20\sqrt{n*0.24}}\right) \Rightarrow \frac{400-8n}{20\sqrt{n*0.24}} = -0.85 \Rightarrow \text{n} = 60$$

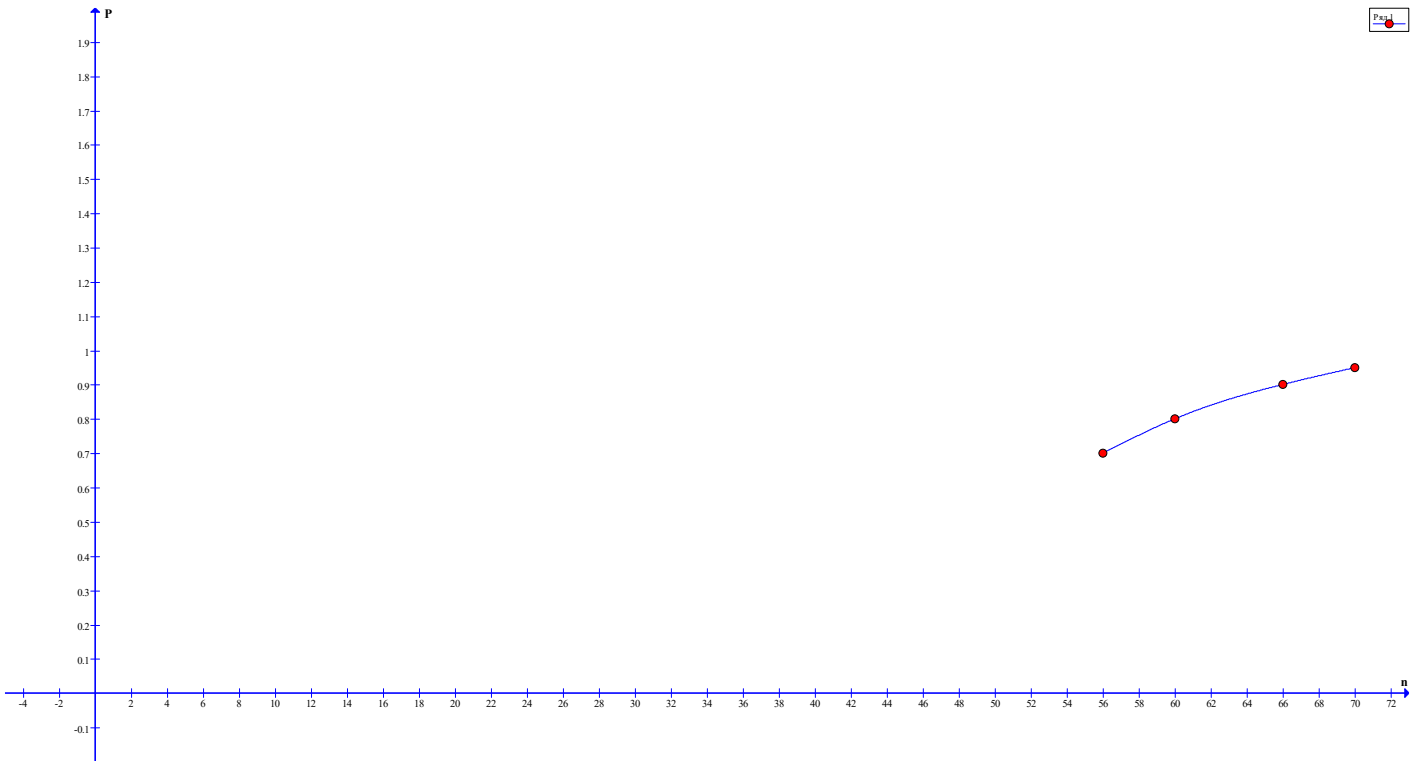
P = 0.9:

$$-0.4 = F\left(\frac{400-8n}{20\sqrt{n*0.24}}\right) \Rightarrow \frac{400-8n}{20\sqrt{n*0.24}} = -1.29 \Rightarrow \text{n} = 66$$

P = 0.95:

$$-0.45 = F\left(\frac{400-8n}{20\sqrt{n*0.24}}\right) \Rightarrow \frac{400-8n}{20\sqrt{n*0.24}} = -1.65 \Rightarrow \text{n} = 70$$

```cpp
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;

int main()
{
    int n;
    cout << "n = ";
    cin >> n;
    cout << endl;

    int r = 8, g = 7, b = 5;
    double p = (double)r / ((double)r + (double)g + (double)b);
    double q = 1 - p;

    double N1 = (long double)r + (long double)g + (long double)b;
    for (int k = 17; k <= n; k++)
    {
        double f = (double)(400 - 8 * k) / ((double)20 * sqrt((double)n * 0.24));
        cout << k << ": " << f << endl;
    }


    return 0;
}
```

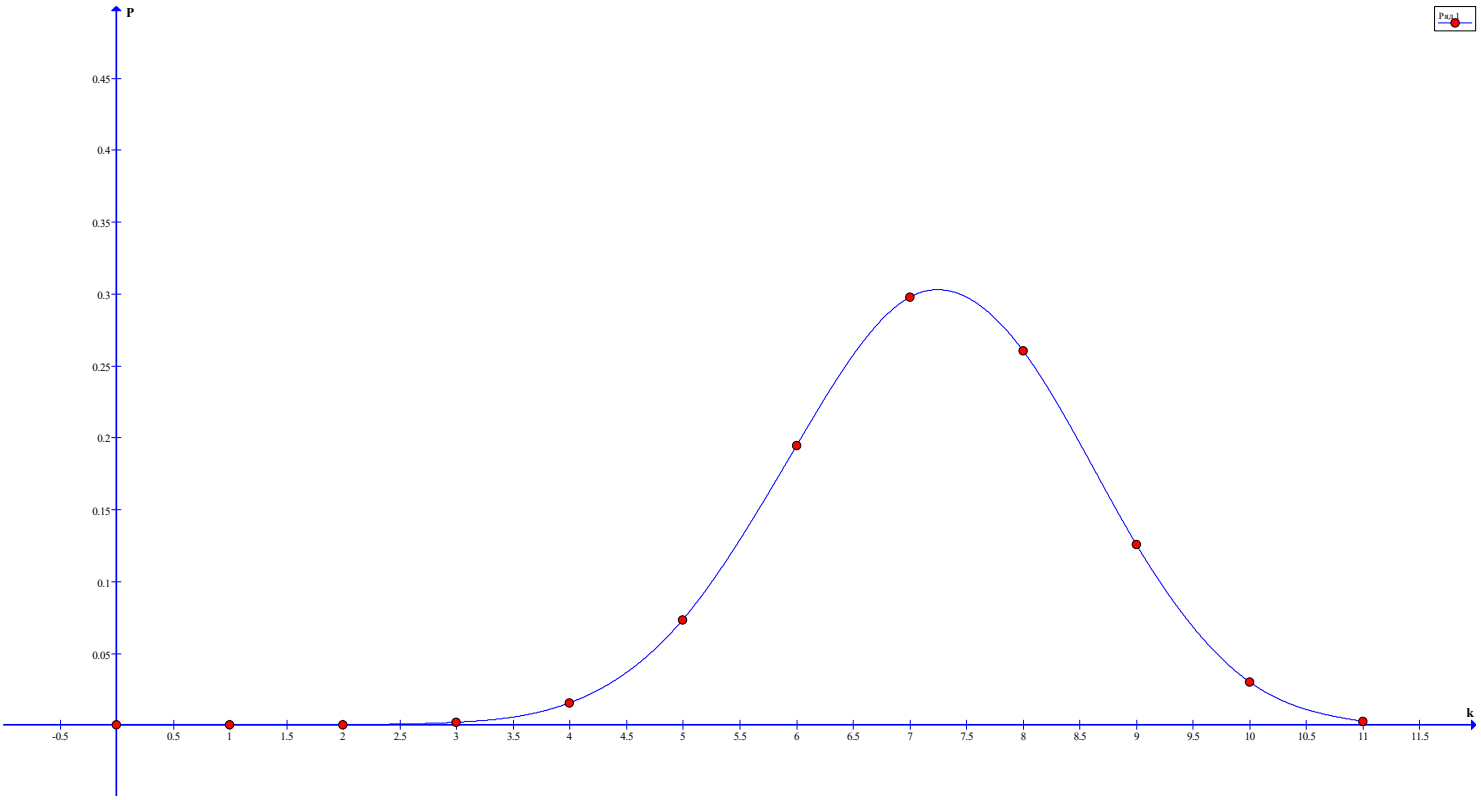


Задача 2. Рассматривается извлечение шаров без возвращения из второй корзины (см. исходные данные к ДЗ №1: R2 = 11, G2 = 10, B2 = 11). Выполняется серия из n=G2+B2=10 +11=21 экспериментов, подсчитывается число k извлечений красных шаров.

1. Построить график вероятности P(k).
2. Построить график функции распределения F(x).
3. Рассчитать математическое ожидание числа извлечённых красных шаров k.
4. Рассчитать дисперсию числа извлечённых красных шаров k.

2.1 $P(k) = \frac{C_{R2}^{k} * C_{N-R2}^{n-k}}{C_{N}^{n}}$

```
P(0): 7.75e-09
P(1): 1.79e-06
P(2): 8.952e-05
P(3): 0.001701
P(4): 0.01531
P(5): 0.07286
P(6): 0.1943
P(7): 0.2974
P(8): 0.2602
P(9): 0.1253
P(10): 0.03007
P(11): 0.002734
```



Программа для вычислений:

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

double comb(int n, int k)
{
    if ((k == 0) || (k == n))
        return 1;
    if ((k == 1) || (k == n - 1))
        return n;

    int numStart;
    if (k > (n / 2))
        numStart = k + 1;
    else
        numStart = (n - k) + 1;

    int denom;
    if (k > (n / 2))
        denom = n - k;
    else
        denom = k;

    int amount;
    if (denom > (n - numStart))
        amount = n - numStart + 1;
    else
        amount = denom;

    double res = 1;
    int divider = 1, multiplier = numStart;
    for (int i = 1; i <= amount; ++i)
    {
        res *= multiplier;
        res /= divider;

        multiplier++;
        divider++;
    }

    return res;
}
```

```cpp
int main()
{
    int r = 11, g = 10, b = 11;
    int N = r + g + b;
    int n = g + b;

    for (int k = 0; k <= r; ++k)
    {
        cout << "P(" << k << "): " << setprecision(4) << comb(r, k) * comb(N - r, n - k) / comb(N, n) << endl;
    }

}
```

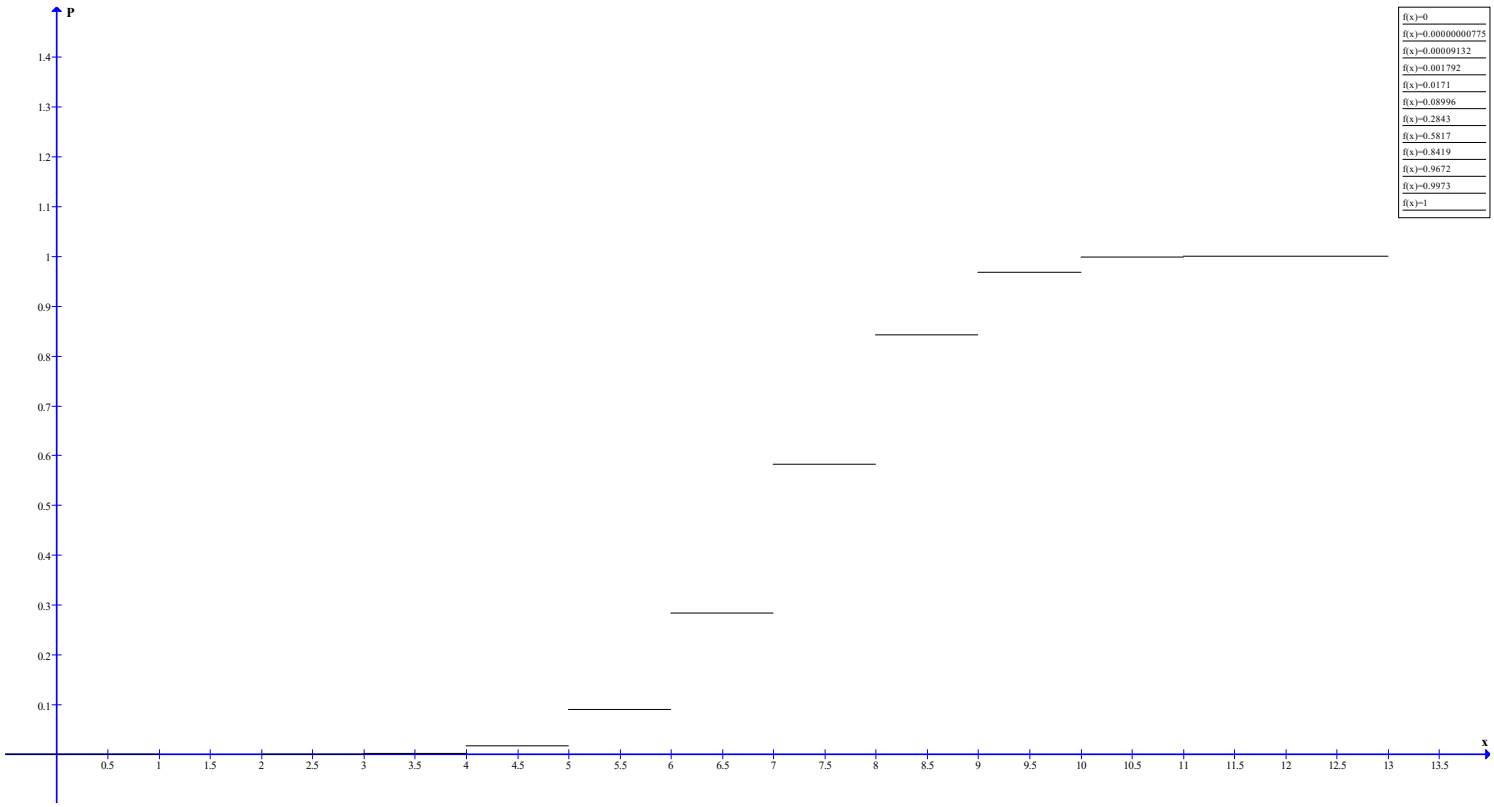
2.2

```
P(0): 7.75e-09 F = 0
P(1): 1.79e-06 F = 7.75e-09
P(2): 8.952e-05 F = 1.798e-06
P(3): 0.001701 F = 9.132e-05
P(4): 0.01531 F = 0.001792
P(5): 0.07286 F = 0.0171
P(6): 0.1943 F = 0.08996
P(7): 0.2974 F = 0.2843
P(8): 0.2602 F = 0.5817
P(9): 0.1253 F = 0.8419
P(10): 0.03007 F = 0.9672
P(11): 0.002734 F = 0.9973

F = 1
```

Программа:

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

double comb(int n, int k)
{
    if ((k == 0) || (k == n))
        return 1;
    if ((k == 1) || (k == n - 1))
        return n;

    int numStart;
    if (k > (n / 2))
        numStart = k + 1;
    else
        numStart = (n - k) + 1;

    int denom;
    if (k > (n / 2))
        denom = n - k;
    else
        denom = k;

    int amount;
    if (denom > (n - numStart))
        amount = n - numStart + 1;
    else
        amount = denom;

    double res = 1;
    int divider = 1, multiplier = numStart;
    for (int i = 1; i <= amount; ++i)
    {
        res *= multiplier;
        res /= divider;

        multiplier++;
        divider++;
    }

    return res;
}
```

```cpp
int main()
{
    int r = 11, g = 10, b = 11;
    int N = r + g + b;
    int n = g + b;

    double F = 0;
    for (int k = 0; k <= r; ++k)
    {
        double P = comb(r, k) * comb(N - r, n - k) / comb(N, n);
        cout << "P(" << k << "): " << setprecision(4) << P << " F = ";

        cout << setprecision(4) << F << endl;
        F += P;
    }

    cout << endl << "F = " << F << endl;

    return 0;
}
```



2.3

```
M = 7.21875
```

Программа:

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

double comb(int n, int k)
{
    if ((k == 0) || (k == n))
        return 1;
    if ((k == 1) || (k == n - 1))
        return n;

    int numStart;
    if (k > (n / 2))
        numStart = k + 1;
    else
        numStart = (n - k) + 1;

    int denom;
    if (k > (n / 2))
        denom = n - k;
    else
        denom = k;

    int amount;
    if (denom > (n - numStart))
        amount = n - numStart + 1;
    else
        amount = denom;

    double res = 1;
    int divider = 1, multiplier = numStart;
    for (int i = 1; i <= amount; ++i)
    {
        res *= multiplier;
        res /= divider;

        multiplier++;
        divider++;
    }

    return res;
}
```

```cpp
int main()
{
    int r = 11, g = 10, b = 11;
    int N = r + g + b;
    int n = g + b;

    double res = 0;
    for (int k = 0; k <= r; k++)
    {
        double P = comb(r, k) * comb(N - r, n - k) / comb(N, n);
        res += P * (double)k;
    }

    cout << "M = " << res << endl;

    return 0;
}
```

2.4 $D(k) = M(k^2) - \big(M(k)\big)^2$

```
D = 1.78344
```

Программа:

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

double comb(int n, int k)
{
    if ((k == 0) || (k == n))
        return 1;
    if ((k == 1) || (k == n - 1))
        return n;

    int numStart;
    if (k > (n / 2))
        numStart = k + 1;
    else
        numStart = (n - k) + 1;

    int denom;
    if (k > (n / 2))
        denom = n - k;
    else
        denom = k;

    int amount;
    if (denom > (n - numStart))
        amount = n - numStart + 1;
    else
        amount = denom;

    double res = 1;
    int divider = 1, multiplier = numStart;
    for (int i = 1; i <= amount; ++i)
    {
        res *= multiplier;
        res /= divider;

        multiplier++;
        divider++;
    }

    return res;
}
```

```cpp
double dispersion(int n, int r, int N)
{
    double res1 = 0, res2 = 0;
    for (int k = 0; k < r; ++k)
    {
        double P = comb(r, k) * comb(N - r, n - k) / comb(N , n );
        res1 += (double)(k * k) * P;
        res2 += (double)(k)* P;
    }

    res2 = pow(res2, 2);
    return (res1 - res2);
}

int main()
{
    int r = 11, g = 10, b = 11;
    int N = r + g + b;
    int n = g + b;

    cout << "D = " << dispersion(n, r, N) << endl;

    return 0;
}
```

Задача 3. Рассматривается извлечение шаров без возвращения из третьей корзины (см. исходные данные к ДЗ №1: R3 = 6, G3 = 9, B3 = 8). Выполняется серия из k экспериментов, которая прекращается, когда извлечены все R3 красных шаров.

1. Рассчитать значения P(k).
2. Рассчитать математическое ожидание числа извлечений k.
3. Рассчитать дисперсию числа извлечений k.

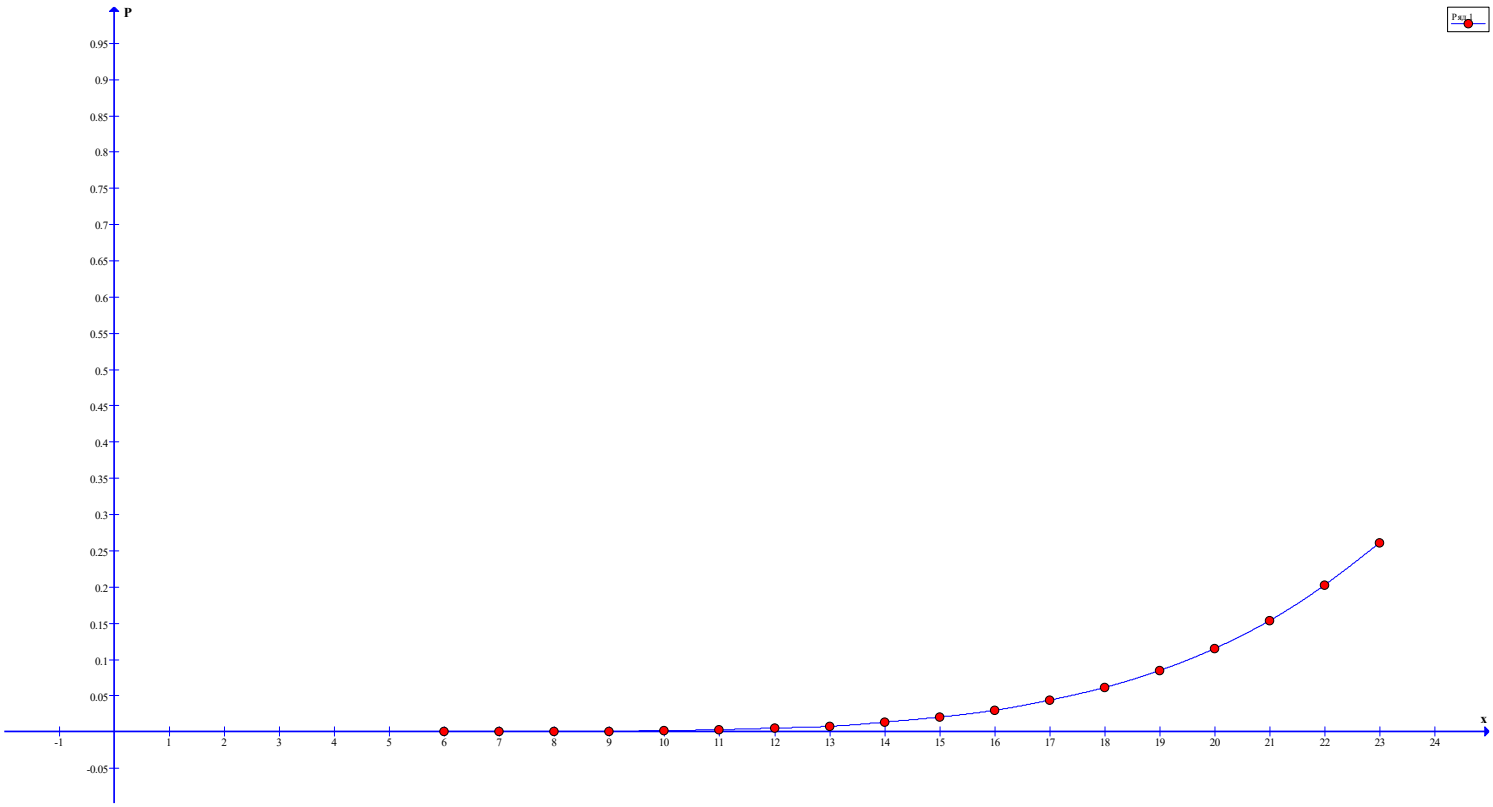
3.1. $P(k) = \frac{C_{R3}^{R3} * C_{N-R3}^{0}}{C_N^{R3}} * C_{k-1}^{k-R3}$

```
P(6): 9.906e-06
P(7): 5.944e-05
P(8): 0.000208
P(9): 0.0005547
P(10): 0.001248
P(11): 0.002496
P(12): 0.004577
P(13): 0.007846
P(14): 0.01275
P(15): 0.01983
P(16): 0.02975
P(17): 0.04327
P(18): 0.0613
P(19): 0.08488
P(20): 0.1152
P(21): 0.1536
P(22): 0.2016
P(23): 0.2609
```

Программа:

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

double comb(int n, int k)
{
    if ((k == 0) || (k == n))
        return 1;
    if ((k == 1) || (k == n - 1))
        return n;

    int numStart;
    if (k > (n / 2))
        numStart = k + 1;
    else
        numStart = (n - k) + 1;

    int denom;
    if (k > (n / 2))
        denom = n - k;
    else
        denom = k;

    int amount;
    if (denom > (n - numStart))
        amount = n - numStart + 1;
    else
        amount = denom;

    double res = 1;
    int divider = 1, multiplier = numStart;
    for (int i = 1; i <= amount; ++i)
    {
        res *= multiplier;
        res /= divider;

        multiplier++;
        divider++;
    }

    return res;
}
```

```cpp
int main()
{
    int r = 6, g = 9, b = 8;
    int N = r + g + b;

    for (int k = r; k <= N; k++)
    {
        double P = comb(k - 1, k - r) / comb(N, r);
        cout << "P(" << k << "): " << setprecision(4) << P << endl;
    }

    return 0;
}
```



3.2

```
M = 20.5714
```

Программа:

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

double comb(int n, int k)
{
    if ((k == 0) || (k == n))
        return 1;
    if ((k == 1) || (k == n - 1))
        return n;

    int numStart;
    if (k > (n / 2))
        numStart = k + 1;
    else
        numStart = (n - k) + 1;

    int denom;
    if (k > (n / 2))
        denom = n - k;
    else
        denom = k;

    int amount;
    if (denom > (n - numStart))
        amount = n - numStart + 1;
    else
        amount = denom;

    double res = 1;
    int divider = 1, multiplier = numStart;
    for (int i = 1; i <= amount; ++i)
    {
        res *= multiplier;
        res /= divider;

        multiplier++;
        divider++;
    }

    return res;
}
```

```cpp
int main()
{
    int r = 6, g = 9, b = 8;
    int N = r + g + b;

    double res = 0;
    for (int k = r; k <= N; k++)
    {
        double P = comb(k - 1, k - r) / comb(N, r);
        res += P * (double)k;
    }

    cout << "M = " << res << endl;

    return 0;
}
```

3.3 4 $D(k) = M(k^2) - \big(M(k)\big)^2$

```
D = 6.2449
```

Программа:

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

double comb(int n, int k)
{
    if ((k == 0) || (k == n))
        return 1;
    if ((k == 1) || (k == n - 1))
        return n;

    int numStart;
    if (k > (n / 2))
        numStart = k + 1;
    else
        numStart = (n - k) + 1;

    int denom;
    if (k > (n / 2))
        denom = n - k;
    else
        denom = k;

    int amount;
    if (denom > (n - numStart))
        amount = n - numStart + 1;
    else
        amount = denom;

    double res = 1;
    int divider = 1, multiplier = numStart;
    for (int i = 1; i <= amount; ++i)
    {
        res *= multiplier;
        res /= divider;

        multiplier++;
        divider++;
    }

    return res;
}
```

```cpp
double dispersion(int r, int N)
{
    double res1 = 0, res2 = 0;
    for (int k = r; k <= N; ++k)
    {
        double P = comb(k - 1, k - r) / comb(N, r);
        res1 += (double)(k * k) * P;
        res2 += (double)(k)*P;
    }

    res2 = pow(res2, 2);
    return (res1 - res2);
}

int main()
{
    int r = 6, g = 9, b = 8;
    int N = r + g + b;

    cout << "D = " << dispersion(r, N) << endl;

    return 0;
}
```