Задача 1

In [1]:
```
import math

x = 200
a = 100
sigma = math.sqrt(2500)
z = (x - a) / sigma
table_value = 0.97725
```

In [2]:
```
def approx(z, b, arr_a):
    summ = 0
    counter = 1
    lambd = 1 / (1 + b * z)
    for a_i in arr_a:
        summ += a_i * math.pow(lambd, counter)
        counter += 1
    result = 1 - (summ / (math.sqrt(2 * math.pi) * math.exp((z ** 2
    return result
```

In [3]:
```
def relative_error(value, table_val):
    return math.fabs((value - table_val) / table_val * 100)
```

In [4]:
```
a_1 = [0.4361836, -0.1201676, 0.937298]
b_1 = 0.33267

a_2 = [0.31938153, -0.35656378, 1.7814779, -1.821256, 1.3302744]
b_2 = 0.2316419

result_1 = approx(z, b_1, a_1)
result_2 = approx(z, b_2, a_2)

err_1 = relative_error(result_1, table_value)
err_2 = relative_error(result_2, table_value)
```

In [5]:
```
print("Первая аппроксимация: ", result_1)
print("Вторая аппроксимация: ", result_2)
print("Первая погрешность: ", err_1)
print("Вторая погрешность: ", err_2)
```

```
Первая аппроксимация:  0.9772411898846421
Вторая аппроксимация:  0.9772499390653653
Первая погрешность:  0.009091521141755398
Вторая погрешность:  6.235316927232897e-06
```

Задача 2

In [6]:
```
table = 1.64485363
p_4 = 0.05
p_5 = 0.95
```

In [7]:
```
def approx_4(p):
    return -(-2 * math.log(math.sqrt(2 * math.pi) * p)) ** 0.5
```

In [8]:
```
def approx_5(p):
    c_5 = [2.515517, 0.802853, 0.010328]
    d_5 = [1.432788, 0.189269, 0.001308]
    t = math.pow((-2 * math.log(1 - p)), 0.5)
    num = 0
    den = 0
    for i in range(len(c_5)):
        num += c_5[i] * math.pow(t, i)
    for j in range(1, len(d_5) + 1):
        den += d_5[j - 1] * math.pow(t, j)
    den += 1
    return t - (num / den)
```

In [9]:
```
approx_4_result = -approx_4(p_4)
approx_5_result = approx_5(p_5)

print("Четвёртая аппроксимация: ", approx_4_result)
print("Пятая аппроксимация: ", approx_5_result)
print("Четвёртая погрешность: ", relative_error(approx_4_result, ta
print("Пятая погрешность: ", relative_error(approx_5_result, table)
```

```
Четвёртая аппроксимация:  2.0380352010450253
Пятая аппроксимация:  1.645211440143815
Четвёртая погрешность:  23.903742185560013
Пятая погрешность:  0.021753312105640844
```

Задача 3

In [10]:
```
n = 15
```

In [11]:
```
def math_ojid(i, n):
    num = i - (3 / 8)
    denum = n + (1 / 4)
    return num / denum
```

In [12]:
```
def approx_6(p):
    return 4.91 * (math.pow(p, 0.14) - math.pow(1 - p, 0.14))
```

In [13]:
```
p_3 = math_ojid(3, 15)
p_7 = math_ojid(7, 15)
p_9 = math_ojid(9, 15)

result_approx_3 = approx_6(p_3)
result_approx_7 = approx_6(p_7)
result_approx_9 = approx_6(p_9)
```

In [14]:
```
print("Матожидание 3-й статистики: ", result_approx_3)
print("Матожидание 7-й статистики: ", result_approx_7)
print("Матожидание 9-й статистики: ", result_approx_9)
```

```
Матожидание 3-й статистики:  -0.9438796215697801
Матожидание 7-й статистики:  -0.16438470056606225
Матожидание 9-й статистики:  0.16438470056606225
```

Задача 4

In [15]:
```
l = 5
r = 10
x_1 = 7
x_2 = 6
x_3 = 8
```

In [16]:
```
def F(i, a, b):
    return (i - a) / (b - a)
```

In [17]:
```
pogresh_7 = F(x_1, l, r)
pogresh_6 = F(x_2, l, r)
pogresh_8 = F(x_3, l, r)
print("Погрешность не превысит 7 ед.: ", pogresh_7)
x_95 = ((r - l) * 0.05) + l
print("Погрешность измерения: ", x_95)
print("Погрешность в интервале 6-8: ", pogresh_8 - pogresh_6)
```

```
Погрешность не превысит 7 ед.:  0.4
Погрешность измерения:  5.25
Погрешность в интервале 6-8:  0.39999999999999997
```

Задача 5

In [18]:
```
median = 1000
moda = 400
less_than = 2000
# approximation 3
```

In [19]:
```
def get_z(a, b, x):
    num = math.log(x) - a
    return num / b
```

In [20]:
```
def approx_3(z):
    if (z <= 0): return -1
    degree = math.pow(((z + 1.5774) / 2.0637), 2.34) * -1
    return 1 - 0.852 * math.exp(degree)
```

In [21]:
```
a = math.log(median)
b = math.sqrt(a - math.log(moda))
z = get_z(a, b, less_than)
approx_3_result = approx_3(z)

print("Вероятность работы меньше 2000 часов: ", approx_3_result)
```

```
Вероятность работы меньше 2000 часов:  0.7656438405458159
```

Задача 6

In [22]:
```
alpha = 3
beta = math.pow(10, 5)
x = 300000
```

In [23]:
```
def gamma_distribution(x, a, b):
    sum = 0
    x_b = x / b
    for i in range(0, a + 1):
        sum += math.pow(x_b, i) * (1 / math.factorial(i))
    return 1 - (math.exp(-x_b) * sum)
```

In [24]:
```
p = gamma_distribution(x, alpha, beta)
print("Вероятность непревышения 300000ч.: ", p)
```

```
Вероятность непревышения 300000ч.:  0.35276811121776874
```

Задача 7

In [25]:
```
alpha = 2
beta = 3
a = 5
b = 6
```

In [26]:
```
def get_moda(alpha, beta):
    return alpha * math.pow(1 - (1 / beta), 1 / beta)
```

In [27]:
```
def veibulla_distribution(x, a, b):
    return 1 - math.exp(-math.pow((x / a), b))
```

In [28]:
```
moda = get_moda(alpha, beta)
f_6 = veibulla_distribution(b, alpha, beta)
f_5 = veibulla_distribution(a, alpha, beta)
result = f_6 - f_5
print("Мода: ", moda)
print("Вероятность наработки в интервале 5-6: ", result)
```

```
Мода:  1.7471609294725978
Вероятность наработки в интервале 5-6:  1.6373583355822063e-07
```

Задача 8

In [29]:
```
a = 4
x = 3
```

In [30]:
```
def reley_distribution(x, a):
    return 1 - math.exp(-math.pow(x, 2) / (2 * math.pow(a, 2)))
```

In [31]:
```
p = reley_distribution(x, a)
x_95 = math.sqrt(math.log(0.05) * -32)

print("Вероятность непревышения x=3: ", p)
print("95%-я квантиль: ", x_95)
```

```
Вероятность непревышения x=3:  0.24516039801099265
95%-я квантиль:  9.790987322723266
```

Задача 9

In [32]:
```
lmbd = math.pow(10, -5)
a = 1200
b = 1500
over_p = 0.8
lmbd_2 = lmbd * 0.5
```

In [33]:
```
def exp_destribution(x, a):
    if (x < 0): raise ArithmeticError("x must be greater or equal 0
    return 1 - math.exp(-x / a)
```

1) Наработка превысит 1000ч

In [34]:
```
p_1 = 1 - exp_destribution(1000, 1 / lmbd)
print("Наработка превысит 1000ч: ", p_1)
```

```
Наработка превысит 1000ч:  0.9900498337491681
```

2) Наработка в интервале 1200-1500ч

In [35]:
```
p_2 = exp_destribution(1500, 1 / lmbd) - exp_destribution(1200, 1 /
print("Наработка в интервале 1200-1500: ", p_2)
```

```
Наработка в интервале 1200-1500:  0.0029597732588678705
```

3) Наработка с вероятностью 0.8

In [36]:
```
y = -math.log(over_p) / lmbd
print("Наработка с вероятность 0.8: ", y)
```

```
Наработка с вероятность 0.8:  22314.35513142097
```

4) При снижении интенствности отказов

In [37]:
```
y_2 = -math.log(over_p) / lmbd_2
print("Наработка с вероятность 0.8: ", y_2)
```

```
Наработка с вероятность 0.8:  44628.71026284194
```