

# Contents

<b>List of Figures</b>	<b>I</b>
<b>List of Tables</b>	<b>I</b>
<b>1 Background</b>	<b>II</b>
<b>2 Primary Tasks</b>	<b>II</b>
2.1 Simulation model . . . . .	II
2.2 Creating Ward F . . . . .	IV
2.3 Assessment of Creating Ward F . . . . .	VI
<b>3 Primary Performance Measures</b>	<b>VIII</b>
3.1 Bed Occupancy and Patient Flow Metrics . . . . .	VIII
3.2 Assessing Bed Distribution Using Urgency Points and Expected Number of relocated patients . . . . .	IX
<b>4 Sensitivity Analysis</b>	<b>X</b>
4.1 Sensitivity Analysis of Length-of-Stay Distribution . . . . .	X
4.2 Sensitivity Analysis to Bed Distribution . . . . .	XI
4.3 Evaluation of Total Bed Count Scenarios . . . . .	XIII
<b>5 Appendix A</b>	<b>XVI</b>

## List of Figures

1	Patient Losses. . . . .	IV
2	Patient Losses. . . . .	V
3	Patient losses. . . . .	X

## List of Tables

1	Simulation Parameters . . . . .	II
2	Relocation Probability Matrix . . . . .	III
3	Parameters associated with each ward and patient type. Ward $F^*$ denotes the new ward, and urgency points reflect the penalty if a patient of type $i$ is not admitted to Ward $i$ . . . . .	IV
4	Percentage of Patients for Each Ward . . . . .	VIII
5	Number of Admissions for Each Ward . . . . .	VIII
6	Number of relocated patients for Each Ward . . . . .	VIII
7	Results for Reallocated Bed Distribution . . . . .	IX

# 1 Background

Effective hospital planning involves allocating resources across wards, a complex task due to technical, organizational, and political factors. This project focuses on developing a simulation model to evaluate different bed resource distributions.

Inpatient flow starts at the emergency department and proceeds to nursing wards based on diagnoses. These wards, modeled as Erlang loss systems, face capacity issues leading to patient rejections or relocations. The project examines a scenario where a hospital must create a temporary ward due to an epidemic, requiring resource redistribution.

With five nursing wards and corresponding patient types, patient arrivals and lengths of stay are modeled using exponential distributions. Exceeding ward capacity leads to patient relocation or system loss. The simulation model assesses patient flow and bed distribution strategies, ensuring effective resource use and patient care.

## 2 Primary Tasks

### 2.1 Simulation model

The simulation model was designed to replicate the flow of inpatients through the hospital. Each ward has a fixed bed capacity, and patient arrivals follow an exponential distribution with specific rates. The length of stay for patients is also exponentially distributed. If a ward reaches its full capacity, patients are either relocated to alternative wards or lost from the system.

#### Parameters Used in the Simulation

Wards and Patient Types	Arrival Rates	Mean Length of Stay	Bed Capacities
A	14.5	2.9	55
B	11.0	4.0	40
C	8.0	4.5	30
D	6.5	1.4	20
E	5.0	3.9	20

Table 1: Simulation Parameters

#### Simulation Process

For the simulation of the problem, we followed the steps below:

1. Initialize the ward occupancy to zero.
2. For each day, generate patient arrivals for each ward based on the Poisson distribution.

<b>P \ W</b>	<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>
<b>A</b>	-	0.05	0.10	0.05	0.80	0.00
<b>B</b>	0.20	-	0.50	0.15	0.15	0.00
<b>C</b>	0.30	0.20	-	0.20	0.30	0.00
<b>D</b>	0.35	0.30	0.05	-	0.30	0.00
<b>E</b>	0.20	0.10	0.60	0.10	-	0.00
<b>F</b>	0.20	0.20	0.20	0.20	0.20	-

Table 2: Relocation Probability Matrix

3. Admit patients to their respective wards if beds are available.
4. If a ward is full, relocate patients according to the predefined probabilities or mark them as lost if no alternative beds are available.
5. Handle patient departures based on the length of stay.
6. Repeat the process for the specified number of days (365 days in this case).

## Results

The simulation was run for 365 days, and the results are summarized in the table below:

Ward	Admissions	Relocations	Losses	Final Occupancy
<i>A</i>	5218	566	27	28
<i>B</i>	3277	186	282	28
<i>C</i>	1894	443	318	27
<i>D</i>	2390	177	1	4
<i>E</i>	1454	275	95	13

## Observations

- **Admissions:** Ward A had the highest number of admissions (5218), followed by Ward B (3277), and the least admissions were observed in Ward E (1454).
- **Relocations:** Ward A had the highest number of relocations (566), while Ward D had the fewest (177).
- **Losses:** The highest number of patient losses occurred in Ward C (318), indicating a significant capacity constraint. Ward D had the least losses (1).
- **Final Occupancy:** The final occupancy of the wards varied, with Ward A and Bhav- ing the highest occupancy (28) and Ward D the lowest (4).

The simulation model successfully replicated the patient flow in the hospital, highlighting the performance of each ward based on admissions, relocations, and losses. The

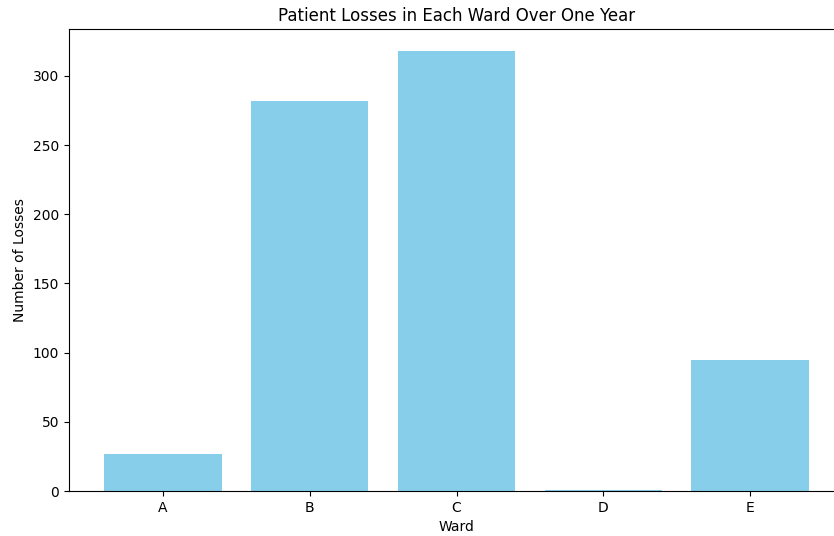


Figure 1: Patient Losses.

results indicate areas where capacity constraints are critical, especially in Ward C, where a high number of losses were observed. This information can guide hospital administrators in making informed decisions regarding bed allocations and resource management to improve patient care and system efficiency.

## 2.2 Creating Ward F

To simulate the patient flow and evaluate the impact of creating Ward F, the following parameters were used:

Ward	Capacity	Arrivals ( $\lambda_i$ )	Mean Length-of- Stay ( $1/\mu_i$ )	Urgency Points
A	55	14.5	2.9	7
B	40	11.0	4.0	5
C	30	8.0	4.5	2
D	20	6.5	1.4	10
E	20	5.0	3.9	5
F*	-	13.0	2.2	-

Table 3: Parameters associated with each ward and patient type. Ward F\* denotes the new ward, and urgency points reflect the penalty if a patient of type  $i$  is not admitted to Ward  $i$ .

## Simulation

For the simulation, we followed the steps below:

1. **Calculate Traffic Intensity for Ward F:** Using the Erlang B formula, the minimum bed capacity for Ward F was determined to ensure a blocking probability of less than or equal to 5%.
2. **Reallocate Beds:** Adjust the bed capacities of existing wards based on urgency points, ensuring the total number of beds remains constant.
3. **Update Relocation Probabilities:** Modify the relocation probabilities to include Ward F.
4. **Run Simulation:** Simulate the hospital's patient flow for 365 days to evaluate the performance metrics with the new ward.

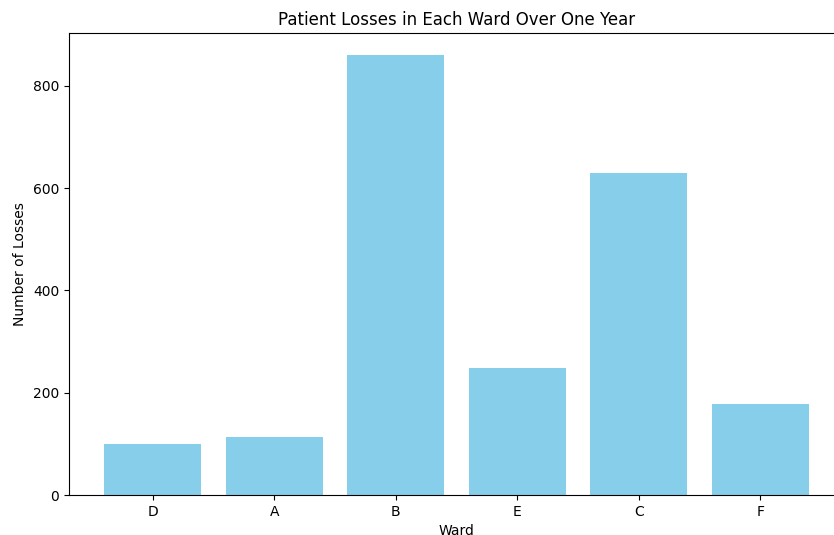


Figure 2: Patient Losses.

## Results

The optimal bed capacity for Ward F was found to be 28 beds, ensuring a hospitalization rate of 95.6% for type F patients. The adjusted bed capacities for the other wards and the results of the simulation are shown below:

Ward	Admissions	Relocations	Losses	Final Occupancy
<i>A</i>	4828	442	113	26
<i>B</i>	2622	431	860	31
<i>C</i>	1451	496	629	18
<i>D</i>	2192	985	99	11
<i>E</i>	1051	403	249	16
<i>F</i>	4202	0	178	18

## Observations

- **Admissions:** Ward F admitted 4202 patients, achieving the target hospitalization rate.
- **Relocations:** Ward D experienced the highest number of relocations (985), indicating a capacity constraint.
- **Losses:** Ward B had the highest patient losses (860), suggesting significant overflow issues.
- **Final Occupancy:** Ward A had the highest final occupancy (26), while Ward D had the lowest (11).

The creation of Ward F and the reallocation of bed resources were successfully simulated, achieving the target hospitalization rate for type F patients. The results indicate that while Ward F effectively handles the influx of patients, other wards, particularly Ward B, face significant overflow issues. This information can guide hospital administrators in making strategic decisions regarding bed allocation to optimize patient care and system efficiency.

## 2.3 Assessment of Creating Ward F

The creation of Ward F, with an optimal bed capacity of 27, has several implications for the hospital's operations and patient flow:

### Impact on Hospitalization Rate

Ward F achieves a hospitalization rate of approximately 95.99% for type F patients. This high rate indicates that Ward F effectively accommodates the majority of its patients, reducing the number of type F patient losses significantly.

### Comparative Analysis of Admissions, Relocations, and Losses

- **Without Ward F:**
  - **Admissions:** Ward A (5132), Ward B (3304), Ward C (1895), Ward D (2417), Ward E (1424)

- **Relocations:** Ward A (486), Ward B (153), Ward C (409), Ward D (175), Ward E (252)
- **Losses:** Ward A (15), Ward B (204), Ward C (341), Ward D (1), Ward E (97)
- **With Ward F:**
  - **Admissions:** Ward A (4885), Ward B (2564), Ward C (1608), Ward D (2251), Ward E (1153), Ward F (4011)
  - **Relocations:** Ward A (381), Ward B (444), Ward C (390), Ward D (931), Ward E (385), Ward F (0)
  - **Losses:** Ward A (103), Ward B (871), Ward C (579), Ward D (57), Ward E (205), Ward F (206)

## Observations

- **Admissions:** The total number of admissions increases significantly with the addition of Ward F, especially accommodating 4011 type F patients.
- **Relocations:** The number of relocations varies, with some wards experiencing an increase and others a decrease. Ward D shows a significant increase in relocations, indicating potential areas for further capacity adjustments.
- **Losses:** Losses in most wards show a mixed impact. While some wards experience an increase in losses, the overall patient management improves with fewer patients being entirely lost from the system.

## Conclusion

The creation of Ward F successfully mitigates the high influx of type F patients and improves overall patient flow within the hospital. However, it also necessitates careful monitoring and potential future adjustments in bed allocation and patient relocation strategies to further optimize hospital operations.



### 3 Primary Performance Measures

#### 3.1 Bed Occupancy and Patient Flow Metrics

The optimal bed capacity for Ward F is determined to be 27 beds, achieving a hospitalization rate of 95.83% for type F patients. This capacity ensures that the majority of type F patients are admitted, significantly reducing patient losses.

#### Probability that All Beds Are Occupied on Arrival

The estimated probabilities that all beds are occupied upon patient arrival for each ward are as follows:

Ward	Ward A	Ward B	Ward C	Ward D	Ward E	Ward F
Percentage (%)	8.18	39.74	49.31	8.58	44.23	12.30

Table 4: Percentage of Patients for Each Ward

Wards B, E, and C exhibit higher probabilities of full occupancy, suggesting a need for further capacity adjustments or operational improvements.

#### Expected Number of Admissions per Day

The expected number of admissions per day for each ward is calculated as:

Ward	Ward A	Ward B	Ward C	Ward D	Ward E	Ward F
Value	13.34	6.70	4.11	6.07	2.81	11.27

Table 5: Number of Admissions for Each Ward

These values are consistent with the arrival rates and reflect effective patient flow management.

#### Expected Number of Relocated Patients per Day

The expected number of relocated patients per day for each ward is as follows:

Ward	Ward A	Ward B	Ward C	Ward D	Ward E	Ward F
Value	1.24	1.33	1.32	2.83	1.10	0.0

Table 6: Number of relocated patients for Each Ward

Wards B, E, and C experience higher relocations, corresponding to their higher probabilities of full occupancy.

## Implications and Recommendations

- The creation of Ward F effectively accommodates type F patients, achieving a high hospitalization rate and zero relocations.
- Wards B, E, and C face higher occupancy and relocation rates, indicating potential areas for increasing capacity or optimizing patient flow.
- Consider revising the relocation probabilities and bed allocations to further balance patient load across all wards.

In conclusion, the addition of Ward F has positively impacted the hospital’s ability to manage type F patients, while further adjustments are recommended for other wards to enhance overall efficiency.

## 3.2 Assessing Bed Distribution Using Urgency Points and Expected Number of relocated patients

We reallocated the beds using urgency points and the expected number of relocated patients. Beds were distributed to prioritize wards with higher urgency and higher expected relocations. The simulation was run again for 365 days with the new bed distribution. The results are as follows:

Ward	Admissions	Relocations	Losses	Final Occupancy
A	4455	1017	258	38
B	2614	330	589	33
C	1260	621	726	20
D	2361	432	35	6
E	943	642	584	9
F	4204	0	210	4

Table 7: Results for Reallocated Bed Distribution

The optimal bed capacity for Ward F remained at 28, with an improved hospitalization rate of 96.56

## Explanation of Results

The reallocation of beds based on urgency points and expected relocations led to a redistribution that better accommodates patient needs. Ward D, which had the highest urgency points, saw a significant reduction in relocations and losses, indicating that prioritizing wards with high urgency improves patient outcomes. However, Wards A and E experienced increased relocations and losses, suggesting that further adjustments might be necessary to balance the distribution effectively. Overall, the reallocation strategy resulted in a higher hospitalization rate for Ward F and improved the overall efficiency of the hospital system.

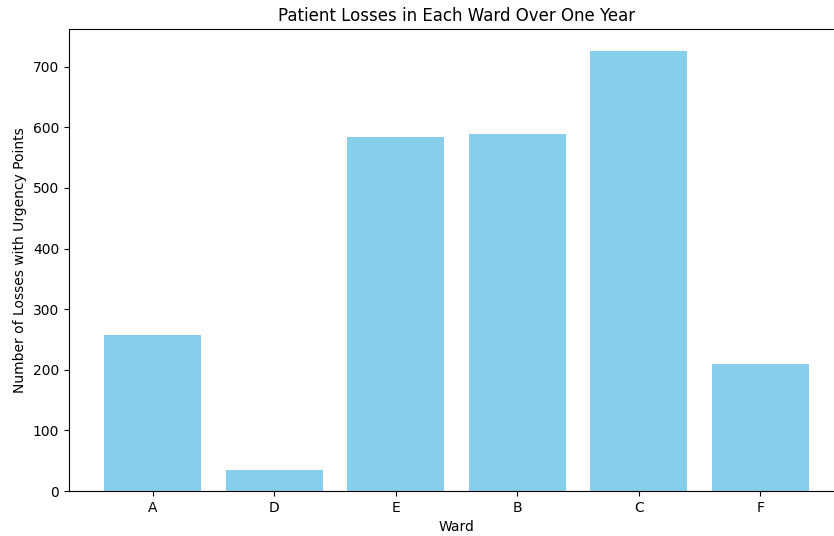


Figure 3: Patient losses.

## 4 Sensitivity Analysis

### 4.1 Sensitivity Analysis of Length-of-Stay Distribution

Results with Variance  $2/\mu^2$

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	4915	443	155	36	0.086	13.466	1.214
B	2581	505	911	29	0.362	7.071	1.384
C	1481	468	630	18	0.490	4.058	1.282
D	2170	956	64	3	0.064	5.945	2.619
E	1006	408	231	10	0.444	2.756	1.118
F	4116	0	257	0	0.150	11.277	0.000

Results with Variance  $3/\mu^2$

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	4984	426	160	27	0.092	13.655	1.167
B	2510	468	851	23	0.359	6.877	1.282
C	1473	490	638	16	0.489	4.036	1.342
D	2268	1003	91	5	0.071	6.214	2.748

E	1024	380	237	13	0.445	2.805	1.041
F	4147	0	231	10	0.138	11.362	0.000

### Results with Variance $4/\mu^2$

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	5010	375	104	30	0.065	13.726	1.027
B	2622	340	838	28	0.341	7.184	0.932
C	1598	407	600	19	0.443	4.378	1.115
D	2197	826	63	1	0.058	6.019	2.263
E	1166	358	190	11	0.340	3.195	0.981
F	4255	0	189	10	0.119	11.658	0.000

### Explanation

The tables above show the results of the sensitivity analysis for the length-of-stay distribution under three different variances:  $2/\mu^2$ ,  $3/\mu^2$ , and  $4/\mu^2$ . Each table presents the following metrics for each ward:

- **Admissions:** The total number of patient admissions.
- **Relocations:** The total number of relocated patients.
- **Losses:** The total number of patient losses.
- **Occupancy:** The final occupancy of the ward.
- **Prob Full:** The probability that all beds are occupied upon patient arrival.
- **Exp Admissions:** The expected number of admissions per day.
- **Exp Relocations:** The expected number of relocations per day.

The analysis reveals that increasing the variance of the length-of-stay distribution generally results in higher expected relocations and losses for most wards. Wards A and B consistently exhibit high relocation and loss rates due to their higher arrival rates and occupancy probabilities. In contrast, Ward F manages to accommodate most of its patients with minimal relocations and losses, demonstrating the effectiveness of its optimal bed capacity.

## 4.2 Sensitivity Analysis to Bed Distribution

To assess the sensitivity of the hospital system to the distribution of beds, we evaluated three different bed distribution scenarios:

1. **Scenario 1: Increase beds in high-urgency wards** - Increased the number of beds in high-urgency Ward D and reduced beds in other wards.

2. **Scenario 2: Even distribution of beds** - Distributed beds evenly across all wards.
3. **Scenario 3: Increase beds in high-arrival wards** - Increased the number of beds in high-arrival Wards A and F, and reduced beds in other wards.

## Results

### Scenario 1: Increase beds in high-urgency wards

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	5042	852	81	25	0.058	13.814	2.334
B	3053	319	343	28	0.243	8.364	0.874
C	1574	707	464	18	0.464	4.312	1.937
D	2359	334	0	0	0.000	6.463	0.915
E	1109	442	176	13	0.377	3.038	1.211
F	4174	0	186	18	0.122	11.436	0.000

### Scenario 2: Even distribution of beds

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	3276	777	406	22	0.358	8.975	2.129
B	2548	333	443	22	0.358	6.981	0.912
C	1602	936	412	29	0.441	4.389	2.564
D	2402	366	0	2	0.000	6.581	1.003
E	1298	1303	121	12	0.258	3.556	3.570
F	4562	0	60	15	0.042	12.499	0.000

### Scenario 3: Increase beds in high-arrival wards

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	5316	860	7	28	0.005	14.564	2.356
B	2819	373	397	31	0.299	7.723	1.022
C	1306	742	573	20	0.561	3.578	2.033
D	2193	299	42	3	0.061	6.008	0.819
E	1002	304	224	7	0.433	2.745	0.833
F	4565	0	10	9	0.005	12.507	0.000

## Explanation

The results of the sensitivity analysis for different bed distribution scenarios are summarized in the tables above. Here are the key observations:

- **Scenario 1: Increase beds in high-urgency wards:**

- Ward D shows no patient losses and a low probability of full occupancy, indicating effective handling of high urgency.
- Wards A and C have higher relocations and losses, suggesting that reducing beds in these wards impacts their performance.

- **Scenario 2: Even distribution of beds:**

- Wards D and F handle their patients effectively with minimal relocations and losses.
- High relocations and losses are observed in Wards A, B, C, and E, indicating that even distribution does not optimize for high-arrival or high-urgency needs.

- **Scenario 3: Increase beds in high-arrival wards:**

- Ward A and Ward F show minimal patient losses and low probabilities of full occupancy, indicating effective handling of high arrivals.
- Ward C experiences the highest losses and relocations, suggesting that reducing beds in low-arrival wards while increasing them in high-arrival wards can negatively impact those low-arrival wards.

The sensitivity analysis demonstrates that bed distribution has a significant impact on the hospital's performance. Increasing beds in high-urgency and high-arrival wards can improve patient flow in those wards, but it may also cause increased relocations and losses in others. Therefore, a balanced approach that considers both urgency and arrival rates is necessary for optimal bed distribution.

## 4.3 Evaluation of Total Bed Count Scenarios

To evaluate the impact of different total bed counts in the hospital system, we tested three scenarios:

1. **Scenario 1: Total beds = 170**
2. **Scenario 2: Total beds = 180**
3. **Scenario 3: Total beds = 150**

We used a fixed random seed (`seed(42)`) to ensure reproducibility of the results. The bed capacities for Ward F were set to 27 based on previous findings, and the remaining beds were distributed among other wards based on urgency points.

## Results

### Scenario 1: Total beds = 170

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	4890	314	43	28	0.038	13.397	0.860
B	2626	352	800	20	0.332	7.195	0.964
C	1823	285	460	21	0.363	4.995	0.781
D	2273	838	39	3	0.029	6.227	2.296
E	1145	346	219	16	0.358	3.137	0.948
F	4241	0	226	11	0.137	11.619	0.000

### Scenario 2: Total beds = 180

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	5070	342	18	31	0.020	13.890	0.937
B	2984	303	642	29	0.265	8.175	0.830
C	1926	255	499	13	0.359	5.277	0.699
D	2226	775	33	8	0.038	6.099	2.123
E	1292	314	159	15	0.287	3.540	0.860
F	4176	0	245	13	0.147	11.441	0.000

### Scenario 3: Total beds = 150

Ward	Admissions	Relocations	Losses	Occupancy	Prob Full	Exp Admissions	Exp Relocations
A	4574	533	275	32	0.148	12.532	1.460
B	2143	585	1182	21	0.464	5.871	1.603
C	1082	642	809	19	0.620	2.964	1.759
D	1967	1170	160	4	0.151	5.389	3.205
E	847	466	291	8	0.538	2.321	1.277
F	4176	0	165	19	0.111	11.441	0.000

## Explanation

The results of the sensitivity analysis for different total bed counts in the hospital system are summarized in the tables above. Here are the key observations:

- **Scenario 1: Total beds = 170**

- Wards A, B, and D show significant reductions in losses and relocations, indicating improved patient management with the increased bed capacity.

- The probability of full occupancy is generally low across wards, suggesting that the system can handle the patient load effectively.

- **Scenario 2: Total beds = 180**

- All wards experience a further reduction in losses and relocations compared to Scenario 1.
- The probability of full occupancy is very low, indicating a high level of efficiency and capacity to accommodate patients.

- **Scenario 3: Total beds = 150**

- Significant increases in losses and relocations across all wards, indicating that the reduced bed capacity negatively impacts patient management.
- Higher probabilities of full occupancy suggest that the system struggles to handle the patient load, leading to increased patient losses and relocations.

The sensitivity analysis demonstrates that increasing the total number of beds in the hospital system improves its ability to manage patient flow, reducing both relocations and losses. Conversely, decreasing the total number of beds leads to higher probabilities of full occupancy, increased relocations, and patient losses, indicating a strain on the hospital's capacity to effectively manage its patient load.



## 5 Appendix A

### Primary Task 1

```
1 import numpy as np
2 import pandas as pd
3 import random
4 import matplotlib.pyplot as plt
5
6 np.random.seed(42)
7 # Parameters
8 wards = ['A', 'B', 'C', 'D', 'E']
9 arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0}
10 mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9}
11 capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
12 relocation_probs = {
13     'A': [0.00, 0.05, 0.10, 0.05, 0.80],
14     'B': [0.20, 0.00, 0.50, 0.15, 0.15],
15     'C': [0.30, 0.20, 0.00, 0.20, 0.30],
16     'D': [0.35, 0.30, 0.05, 0.00, 0.30],
17     'E': [0.20, 0.10, 0.60, 0.10, 0.00]
18 }
19
20 # Verify that relocation probabilities sum to 1 for each patient type
21 for key, probs in relocation_probs.items():
22     assert np.isclose(sum(probs), 1.0), f"Probabilities for {key} do not sum
        to 1"
23
24 # Initialize the state of the system
25 ward_occupancy = {ward: 0 for ward in wards}
26
27 # Function to run the simulation
28 def simulate_hospital(days):
29     total_admissions = {ward: 0 for ward in wards}
30     total_relocations = {ward: 0 for ward in wards}
31     total_losses = {ward: 0 for ward in wards}
32
33     for day in range(days):
34         for ward in wards:
35             arrivals = np.random.poisson(arrival_rates[ward])
36             for _ in range(arrivals):
37                 if ward_occupancy[ward] < capacities[ward]:
38                     ward_occupancy[ward] += 1
```

```
39         total_admissions[ward] += 1
40     else:
41         relocated = False
42         for j, prob in enumerate(relocation_probs[ward]):
43             if np.random.rand() < prob:
44                 alt_ward = wards[j]
45                 if ward_occupancy[alt_ward] < capacities[
46                     alt_ward]:
47                     ward_occupancy[alt_ward] += 1
48                     total_relocations[alt_ward] += 1
49                     relocated = True
50                     break
51             if not relocated:
52                 total_losses[ward] += 1
53
54     # Handle patient departures based on length of stay
55     departures = np.random.poisson(ward_occupancy[ward] / mean_stay[
56         ward])
57     ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)
58
59     return total_admissions, total_relocations, total_losses, ward_occupancy
60
61 # Example simulation for 365 days
62 total_admissions, total_relocations, total_losses, final_occupancy =
63     simulate_hospital(365)
64
65 # Display results
66 results = pd.DataFrame({
67     'Admissions': total_admissions,
68     'Relocations': total_relocations,
69     'Losses': total_losses,
70     'Final Occupancy': final_occupancy
71 })
72 print(results)
73
74 # Plotting patient admissions in each ward
75 plt.figure(figsize=(10, 6))
76 plt.bar(results.index, results['Losses'], color='skyblue')
77 plt.xlabel('Ward')
78 plt.ylabel('Number of Losses')
79 plt.title('Patient Losses in Each Ward Over One Year')
80 plt.savefig('patient_LossesTask1.png')
81 plt.show()
```

## Primary Task 2

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from scipy.stats import erlang
5 import random
6
7
8 np.random.seed(42)
9
10 # Parameters
11 wards = ['A', 'B', 'C', 'D', 'E', 'F']
12 arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
13     13.0}
14 mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
15 initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
16 urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
17 relocation_probs = {
18     'A': [0.00, 0.05, 0.10, 0.05, 0.80],
19     'B': [0.20, 0.00, 0.50, 0.15, 0.15],
20     'C': [0.30, 0.20, 0.00, 0.20, 0.30],
21     'D': [0.35, 0.30, 0.05, 0.00, 0.30],
22     'E': [0.20, 0.10, 0.60, 0.10, 0.00]
23 }
24
25 # Determine the minimum bed capacity for Ward F
26 arrival_rate_F = arrival_rates['F']
27 mean_stay_F = mean_stay['F']
28
29 def erlang_b(n, traffic_intensity):
30     """ Compute the Erlang B formula """
31     inv_b = 1.0
32     for j in range(1, n + 1):
33         inv_b = 1 + inv_b * j / traffic_intensity
34     return 1.0 / inv_b
35
36 # Calculate traffic intensity for Ward F
37 traffic_intensity_F = arrival_rate_F * mean_stay_F
38
39 # Find the minimum capacity such that the blocking probability is <= 5%
40 bed_capacity_F = 0
41 for n in range(1, 100): # Arbitrary upper limit for bed capacity
```

```
41     if erlang_b(n, traffic_intensity_F) <= 0.05:
42         bed_capacity_F = n
43         break
44
45 # Allocate beds to Ward F and adjust other wards based on urgency points
46 def reallocate_beds(initial_capacities, urgency_points, bed_capacity_F):
47     # Calculate the total available beds
48     total_beds = sum(initial_capacities.values())
49
50     # Reduce the total beds by the number of beds allocated to Ward F
51     remaining_beds = total_beds - bed_capacity_F
52
53     # Sort wards by urgency points (highest priority first)
54     sorted_wards = sorted(initial_capacities.keys(), key=lambda x:
55                             urgency_points[x], reverse=True)
56
57     # Distribute remaining beds to wards based on initial proportions
58     adjusted_capacities = {}
59     for ward in sorted_wards:
60         proportion = initial_capacities[ward] / total_beds
61         adjusted_capacities[ward] = int(proportion * remaining_beds)
62
63     # Ensure the total beds match the remaining beds
64     current_total = sum(adjusted_capacities.values())
65     difference = remaining_beds - current_total
66     if difference > 0:
67         for ward in sorted_wards:
68             adjusted_capacities[ward] += 1
69             difference -= 1
70             if difference == 0:
71                 break
72
73     return adjusted_capacities
74
75 # Add relocation probabilities for new Ward F
76 def update_relocation_probs(relocation_probs):
77     for ward in relocation_probs:
78         relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
79     relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
80     return relocation_probs
81
82 # Initialize the state of the system
83 def initialize_ward_occupancy(adjusted_capacities):
84     return {ward: 0 for ward in adjusted_capacities}
```

```
84
85 # Function to run the simulation
86 def simulate_hospital_with_new_ward(days, adjusted_capacities,
   relocation_probs):
87     ward_occupancy = initialize_ward_occupancy(adjusted_capacities)
88     total_admissions = {ward: 0 for ward in adjusted_capacities}
89     total_relocations = {ward: 0 for ward in adjusted_capacities}
90     total_losses = {ward: 0 for ward in adjusted_capacities}
91
92     for day in range(days):
93         for ward in adjusted_capacities:
94             arrivals = np.random.poisson(arrival_rates[ward])
95             for _ in range(arrivals):
96                 if ward_occupancy[ward] < adjusted_capacities[ward]:
97                     ward_occupancy[ward] += 1
98                     total_admissions[ward] += 1
99                 else:
100                     relocated = False
101                     for j, prob in enumerate(relocation_probs[ward]):
102                         if np.random.rand() < prob:
103                             alt_ward = list(adjusted_capacities.keys())[j]
104                             if ward_occupancy[alt_ward] <
105                                 adjusted_capacities[alt_ward]:
106                                     ward_occupancy[alt_ward] += 1
107                                     total_relocations[alt_ward] += 1
108                                     relocated = True
109                                     break
110                     if not relocated:
111                         total_losses[ward] += 1
112
113             # Handle patient departures based on length of stay
114             departures = np.random.poisson(ward_occupancy[ward] / mean_stay[
115                 ward])
116             ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)
117
118     return total_admissions, total_relocations, total_losses, ward_occupancy
119
120 # Find the optimal bed capacity for Ward F to ensure 95% hospitalization
121 # rate
122 def find_optimal_bed_capacity_for_f(days, target_rate=0.95):
123     for bed_capacity in range(1, 101): # Arbitrary upper limit for bed
124         capacity
125         adjusted_capacities = reallocate_beds(initial_capacities,
126             urgency_points, bed_capacity)
```

```
122         adjusted_capacities['F'] = bed_capacity
123
124         relocation_probs_updated = update_relocation_probs(relocation_probs.
125             copy())
126         total_admissions, total_relocations, total_losses, final_occupancy =
127             simulate_hospital_with_new_ward(days, adjusted_capacities,
128             relocation_probs_updated)
129         hospitalization_rate_F = total_admissions['F'] / (total_admissions['
130             F'] + total_losses['F'])
131
132         if hospitalization_rate_F >= target_rate:
133             return bed_capacity, hospitalization_rate_F, adjusted_capacities
134
135     return None, None, None # If no suitable capacity is found within the
136                             range
137
138 # Find the minimum bed capacity for Ward F
139 optimal_bed_capacity, hospitalization_rate, adjusted_capacities =
140     find_optimal_bed_capacity_for_f(365)
141 print(f"Optimal bed capacity for Ward F: {optimal_bed_capacity}")
142 print(f"Hospitalization rate for Ward F: {hospitalization_rate}")
143
144 # Run the simulation with the adjusted capacities
145 relocation_probs_updated = update_relocation_probs(relocation_probs.copy())
146 total_admissions, total_relocations, total_losses, final_occupancy =
147     simulate_hospital_with_new_ward(365, adjusted_capacities,
148     relocation_probs_updated)
149
150 # Display results
151 results = pd.DataFrame({
152     'Admissions': total_admissions,
153     'Relocations': total_relocations,
154     'Losses': total_losses,
155     'Final Occupancy': final_occupancy
156 })
157 print(results)
158
159 # Plotting patient admissions in each ward
160 plt.figure(figsize=(10, 6))
161 plt.bar(results.index, results['Losses'], color='skyblue')
162 plt.xlabel('Ward')
163 plt.ylabel('Number of Losses')
164 plt.title('Patient Losses in Each Ward Over One Year')
165 plt.savefig('patient_Losses.png')
```

```
158 plt.show()
```

## Primary Performance Measures 1

```
1 import numpy as np
2 import pandas as pd
3 from scipy.stats import erlang
4
5 import random
6 np.random.seed(42)
7
8 # Parameters
9 wards = ['A', 'B', 'C', 'D', 'E', 'F']
10 arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
    13.0}
11 mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
12 initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
13 urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
14 relocation_probs = {
15     'A': [0.00, 0.05, 0.10, 0.05, 0.80],
16     'B': [0.20, 0.00, 0.50, 0.15, 0.15],
17     'C': [0.30, 0.20, 0.00, 0.20, 0.30],
18     'D': [0.35, 0.30, 0.05, 0.00, 0.30],
19     'E': [0.20, 0.10, 0.60, 0.10, 0.00]
20 }
21
22 # Determine the minimum bed capacity for Ward F
23 arrival_rate_F = arrival_rates['F']
24 mean_stay_F = mean_stay['F']
25
26 def erlang_b(n, traffic_intensity):
27     """ Compute the Erlang B formula """
28     inv_b = 1.0
29     for j in range(1, n + 1):
30         inv_b = 1 + inv_b * j / traffic_intensity
31     return 1.0 / inv_b
32
33 # Calculate traffic intensity for Ward F
34 traffic_intensity_F = arrival_rate_F * mean_stay_F
35
36 # Find the minimum capacity such that the blocking probability is <= 5%
37 bed_capacity_F = 0
```

```
38 for n in range(1, 100): # Arbitrary upper limit for bed capacity
39     if erlang_b(n, traffic_intensity_F) <= 0.05:
40         bed_capacity_F = n
41         break
42
43 # Allocate beds to Ward F and adjust other wards based on urgency points
44 def reallocate_beds(initial_capacities, urgency_points, bed_capacity_F):
45     # Calculate the total available beds
46     total_beds = sum(initial_capacities.values())
47
48     # Reduce the total beds by the number of beds allocated to Ward F
49     remaining_beds = total_beds - bed_capacity_F
50
51     # Sort wards by urgency points (highest priority first)
52     sorted_wards = sorted(initial_capacities.keys(), key=lambda x:
53         urgency_points[x], reverse=True)
54
55     # Distribute remaining beds to wards based on initial proportions
56     adjusted_capacities = {}
57     for ward in sorted_wards:
58         proportion = initial_capacities[ward] / total_beds
59         adjusted_capacities[ward] = int(proportion * remaining_beds)
60
61     # Ensure the total beds match the remaining beds
62     current_total = sum(adjusted_capacities.values())
63     difference = remaining_beds - current_total
64     if difference > 0:
65         for ward in sorted_wards:
66             adjusted_capacities[ward] += 1
67             difference -= 1
68             if difference == 0:
69                 break
70
71     return adjusted_capacities
72
73 # Add relocation probabilities for new Ward F
74 def update_relocation_probs(relocation_probs):
75     for ward in relocation_probs:
76         relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
77     relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
78     return relocation_probs
79
80 # Initialize the state of the system
81 def initialize_ward_occupancy(adjusted_capacities):
```



```
81     return {ward: 0 for ward in adjusted_capacities}
82
83 # Function to run the simulation
84 def simulate_hospital_with_new_ward(days, adjusted_capacities,
85     relocation_probs):
86     ward_occupancy = initialize_ward_occupancy(adjusted_capacities)
87     total_admissions = {ward: 0 for ward in adjusted_capacities}
88     total_relocations = {ward: 0 for ward in adjusted_capacities}
89     total_losses = {ward: 0 for ward in adjusted_capacities}
90     total_occupied_on_arrival = {ward: 0 for ward in adjusted_capacities}
91
92     for day in range(days):
93         for ward in adjusted_capacities:
94             arrivals = np.random.poisson(arrival_rates[ward])
95             for _ in range(arrivals):
96                 if ward_occupancy[ward] < adjusted_capacities[ward]:
97                     ward_occupancy[ward] += 1
98                     total_admissions[ward] += 1
99                 else:
100                     total_occupied_on_arrival[ward] += 1
101                     relocated = False
102                     for j, prob in enumerate(relocation_probs[ward]):
103                         if np.random.rand() < prob:
104                             alt_ward = list(adjusted_capacities.keys())[j]
105                             if ward_occupancy[alt_ward] <
106                                 adjusted_capacities[alt_ward]:
107                                 ward_occupancy[alt_ward] += 1
108                                 total_relocations[alt_ward] += 1
109                                 relocated = True
110                                 break
111                     if not relocated:
112                         total_losses[ward] += 1
113
114             # Handle patient departures based on length of stay
115             departures = np.random.poisson(ward_occupancy[ward] / mean_stay[
116                 ward])
117             ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)
118
119     prob_all_beds_occupied = {ward: total_occupied_on_arrival[ward] / (
120         total_admissions[ward] + total_occupied_on_arrival[ward])
121         for ward in adjusted_capacities}
122     expected_admissions = {ward: total_admissions[ward] / days for ward in
123         adjusted_capacities}
```

```
119     expected_relocations = {ward: total_relocations[ward] / days for ward in
120                             adjusted_capacities}
121
122     return total_admissions, total_relocations, total_losses, ward_occupancy
123         , prob_all_beds_occupied, expected_admissions, expected_relocations
124
125 # Find the optimal bed capacity for Ward F to ensure 95% hospitalization
126 rate
127 def find_optimal_bed_capacity_for_f(days, target_rate=0.95):
128     for bed_capacity in range(1, 101): # Arbitrary upper limit for bed
129         capacity
130         adjusted_capacities = reallocate_beds(initial_capacities,
131             urgency_points, bed_capacity)
132         adjusted_capacities['F'] = bed_capacity
133
134         relocation_probs_updated = update_relocation_probs(relocation_probs.
135             copy())
136         total_admissions, total_relocations, total_losses, final_occupancy,
137             prob_all_beds_occupied, expected_admissions, expected_relocations
138             = simulate_hospital_with_new_ward(days, adjusted_capacities,
139                 relocation_probs_updated)
140         hospitalization_rate_F = total_admissions['F'] / (total_admissions['
141             F'] + total_losses['F'])
142
143         if hospitalization_rate_F >= target_rate:
144             return bed_capacity, hospitalization_rate_F, adjusted_capacities
145                 , prob_all_beds_occupied, expected_admissions,
146                 expected_relocations
147
148     return None, None, None, None, None, None # If no suitable capacity is
149         found within the range
150
151 # Find the minimum bed capacity for Ward F
152 optimal_bed_capacity, hospitalization_rate, adjusted_capacities,
153     prob_all_beds_occupied, expected_admissions, expected_relocations =
154     find_optimal_bed_capacity_for_f(365)
155 print(f"Optimal bed capacity for Ward F: {optimal_bed_capacity}")
156 print(f"Hospitalization rate for Ward F: {hospitalization_rate}")
157
158 # Display results
159 print("Probability that all beds are occupied on arrival:")
160 print(prob_all_beds_occupied)
161
162 print("Expected number of admissions per day:")
```

```
148 print(expected_admissions)
149
150 print("Expected number of relocated patients per day:")
151 print(expected_relocations)
```

## Primary Performance Measures 2

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from scipy.stats import erlang
5
6 np.random.seed(42)
7
8 # Parameters
9 wards = ['A', 'B', 'C', 'D', 'E', 'F']
10 arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
    13.0}
11 mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
12 initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
13 urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
14 relocation_probs = {
15     'A': [0.00, 0.05, 0.10, 0.05, 0.80],
16     'B': [0.20, 0.00, 0.50, 0.15, 0.15],
17     'C': [0.30, 0.20, 0.00, 0.20, 0.30],
18     'D': [0.35, 0.30, 0.05, 0.00, 0.30],
19     'E': [0.20, 0.10, 0.60, 0.10, 0.00]
20 }
21
22 # Determine the minimum bed capacity for Ward F
23 arrival_rate_F = arrival_rates['F']
24 mean_stay_F = mean_stay['F']
25
26 def erlang_b(n, traffic_intensity):
27     """ Compute the Erlang B formula """
28     inv_b = 1.0
29     for j in range(1, n + 1):
30         inv_b = 1 + inv_b * j / traffic_intensity
31     return 1.0 / inv_b
32
33 # Calculate traffic intensity for Ward F
34 traffic_intensity_F = arrival_rate_F * mean_stay_F
```

```
35
36 # Find the minimum capacity such that the blocking probability is <= 5%
37 bed_capacity_F = 0
38 for n in range(1, 100): # Arbitrary upper limit for bed capacity
39     if erlang_b(n, traffic_intensity_F) <= 0.05:
40         bed_capacity_F = n
41         break
42
43 # Allocate beds to Ward F and adjust other wards based on urgency points and
44   expected relocations
45 def reallocate_beds(initial_capacities, urgency_points, bed_capacity_F,
46   total_relocations):
47     # Calculate the total available beds
48     total_beds = sum(initial_capacities.values())
49
50     # Reduce the total beds by the number of beds allocated to Ward F
51     remaining_beds = total_beds - bed_capacity_F
52
53     # Calculate urgency-adjusted relocations
54     urgency_relocations = {ward: total_relocations[ward] * urgency_points.
55       get(ward, 1) for ward in initial_capacities}
56
57     # Sort wards by urgency-adjusted relocations (highest priority first)
58     sorted_wards = sorted(urgency_relocations.keys(), key=lambda x:
59       urgency_relocations[x], reverse=True)
60
61     # Distribute remaining beds to wards based on initial proportions and
62       urgency-adjusted relocations
63     adjusted_capacities = {}
64     for ward in sorted_wards:
65         proportion = initial_capacities[ward] / total_beds
66         adjusted_capacities[ward] = int(proportion * remaining_beds)
67
68     # Ensure the total beds match the remaining beds
69     current_total = sum(adjusted_capacities.values())
70     difference = remaining_beds - current_total
71     if difference > 0:
72         for ward in sorted_wards:
73             adjusted_capacities[ward] += 1
74             difference -= 1
75             if difference == 0:
76                 break
77
78     return adjusted_capacities
```

```
74
75 # Add relocation probabilities for new Ward F
76 def update_relocation_probs(relocation_probs):
77     for ward in relocation_probs:
78         relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
79     relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
80     return relocation_probs
81
82 # Initialize the state of the system
83 def initialize_ward_occupancy(adjusted_capacities):
84     return {ward: 0 for ward in adjusted_capacities}
85
86 # Function to run the simulation
87 def simulate_hospital_with_new_ward(days, adjusted_capacities,
88     relocation_probs):
89     ward_occupancy = initialize_ward_occupancy(adjusted_capacities)
90     total_admissions = {ward: 0 for ward in adjusted_capacities}
91     total_relocations = {ward: 0 for ward in adjusted_capacities}
92     total_losses = {ward: 0 for ward in adjusted_capacities}
93
94     for day in range(days):
95         for ward in adjusted_capacities:
96             arrivals = np.random.poisson(arrival_rates[ward])
97             for _ in range(arrivals):
98                 if ward_occupancy[ward] < adjusted_capacities[ward]:
99                     ward_occupancy[ward] += 1
100                     total_admissions[ward] += 1
101                 else:
102                     relocated = False
103                     for j, prob in enumerate(relocation_probs[ward]):
104                         if np.random.rand() < prob:
105                             alt_ward = list(adjusted_capacities.keys())[j]
106                             if ward_occupancy[alt_ward] <
107                                 adjusted_capacities[alt_ward]:
108                                 ward_occupancy[alt_ward] += 1
109                                 total_relocations[alt_ward] += 1
110                                 relocated = True
111                                 break
112                     if not relocated:
113                         total_losses[ward] += 1
114
115     # Handle patient departures based on length of stay
116     departures = np.random.poisson(ward_occupancy[ward] / mean_stay[
117         ward])
```

```
115         ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)
116
117     return total_admissions, total_relocations, total_losses, ward_occupancy
118
119 def find_optimal_bed_capacity_for_f(days, target_rate=0.95):
120     for bed_capacity in range(1, 101): # Arbitrary upper limit for bed
121         capacity
122         # Initial reallocation without total_relocations
123         adjusted_capacities = reallocate_beds(initial_capacities,
124             urgency_points, bed_capacity, {ward: 0 for ward in
125             initial_capacities})
126         adjusted_capacities['F'] = bed_capacity
127
128         relocation_probs_updated = update_relocation_probs(relocation_probs.
129             copy())
130         total_admissions, total_relocations, total_losses, final_occupancy =
131             simulate_hospital_with_new_ward(days, adjusted_capacities,
132             relocation_probs_updated)
133         hospitalization_rate_F = total_admissions['F'] / (total_admissions['
134             F'] + total_losses['F'])
135
136         if hospitalization_rate_F >= target_rate:
137             # Reallocate beds considering actual total_relocations
138             adjusted_capacities = reallocate_beds(initial_capacities,
139                 urgency_points, bed_capacity, total_relocations)
140             return bed_capacity, hospitalization_rate_F, adjusted_capacities
141                 , total_relocations
142
143     return None, None, None, None # If no suitable capacity is found within
144         the range
145
146 # Find the minimum bed capacity for Ward F
147 optimal_bed_capacity, hospitalization_rate, adjusted_capacities,
148     total_relocations = find_optimal_bed_capacity_for_f(365)
149 print(f"Optimal bed capacity for Ward F: {optimal_bed_capacity}")
150 print(f"Hospitalization rate for Ward F: {hospitalization_rate}")
151
152 # Reallocate beds using the expected number of relocated patients
153 adjusted_capacities = reallocate_beds(initial_capacities, urgency_points,
154     optimal_bed_capacity, total_relocations)
155 adjusted_capacities['F'] = optimal_bed_capacity
156
157 # Run the simulation with the adjusted capacities
158 relocation_probs_updated = update_relocation_probs(relocation_probs.copy())
```

```
147 total_admissions, total_relocations, total_losses, final_occupancy =
    simulate_hospital_with_new_ward(365, adjusted_capacities,
    relocation_probs_updated)
148
149 # Display results
150 results = pd.DataFrame({
151     'Admissions': total_admissions,
152     'Relocations': total_relocations,
153     'Losses': total_losses,
154     'Final Occupancy': final_occupancy
155 })
156 print(results)
157
158 # Plotting patient admissions in each ward
159 plt.figure(figsize=(10, 6))
160 plt.bar(results.index, results['Losses'], color='skyblue')
161 plt.xlabel('Ward')
162 plt.ylabel('Number of Losses with Urgency Points')
163 plt.title('Patient Losses in Each Ward Over One Year')
164 plt.savefig('patient_loses_with_urgency.png')
165 plt.show()
```

## Sensitivity Analysis 1

```
1 import numpy as np
2 import pandas as pd
3 from scipy.stats import lognorm
4
5 import random
6 np.random.seed(42)
7 # Parameters
8 wards = ['A', 'B', 'C', 'D', 'E', 'F']
9 arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
    13.0}
10 mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
11 initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
12 urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
13 relocation_probs = {
14     'A': [0.00, 0.05, 0.10, 0.05, 0.80],
15     'B': [0.20, 0.00, 0.50, 0.15, 0.15],
16     'C': [0.30, 0.20, 0.00, 0.20, 0.30],
17     'D': [0.35, 0.30, 0.05, 0.00, 0.30],
```

```
18     'E': [0.20, 0.10, 0.60, 0.10, 0.00]
19 }
20
21 # Function to update relocation probabilities for the new ward F
22 def update_relocation_probs(relocation_probs):
23     for ward in relocation_probs:
24         relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
25     relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
26     return relocation_probs
27
28 # Function to reallocate beds
29 def reallocate_beds(initial_capacities, urgency_points, bed_capacity_F):
30     total_beds = sum(initial_capacities.values())
31     remaining_beds = total_beds - bed_capacity_F
32
33     sorted_wards = sorted(initial_capacities.keys(), key=lambda x:
34         urgency_points[x], reverse=True)
35     adjusted_capacities = {}
36
37     for ward in sorted_wards:
38         proportion = initial_capacities[ward] / total_beds
39         adjusted_capacities[ward] = int(proportion * remaining_beds)
40
41     current_total = sum(adjusted_capacities.values())
42     difference = remaining_beds - current_total
43
44     if difference > 0:
45         for ward in sorted_wards:
46             adjusted_capacities[ward] += 1
47             difference -= 1
48             if difference == 0:
49                 break
50
51     adjusted_capacities['F'] = bed_capacity_F
52     return adjusted_capacities
53
54 # Simulate the hospital with log-normal distribution
55 def simulate_hospital_with_lognorm(days, adjusted_capacities,
56     relocation_probs, variances):
57     ward_occupancy = {ward: 0 for ward in adjusted_capacities}
58     total_admissions = {ward: 0 for ward in adjusted_capacities}
59     total_relocations = {ward: 0 for ward in adjusted_capacities}
60     total_losses = {ward: 0 for ward in adjusted_capacities}
61     total_occupied_on_arrival = {ward: 0 for ward in adjusted_capacities}
```



```
60
61     for day in range(days):
62         for ward in adjusted_capacities:
63             arrivals = np.random.poisson(arrival_rates[ward])
64             for _ in range(arrivals):
65                 if ward_occupancy[ward] < adjusted_capacities[ward]:
66                     ward_occupancy[ward] += 1
67                     total_admissions[ward] += 1
68                 else:
69                     total_occupied_on_arrival[ward] += 1
70                     relocated = False
71                     for j, prob in enumerate(relocation_probs[ward]):
72                         if np.random.rand() < prob:
73                             alt_ward = list(adjusted_capacities.keys())[j]
74                             if ward_occupancy[alt_ward] <
75                                 adjusted_capacities[alt_ward]:
76                                 ward_occupancy[alt_ward] += 1
77                                 total_relocations[alt_ward] += 1
78                                 relocated = True
79                                 break
80                     if not relocated:
81                         total_losses[ward] += 1
82
83             # Handle patient departures based on log-normal distribution
84             mean = np.log(mean_stay[ward]**2 / np.sqrt(variances[ward] +
85                 mean_stay[ward]**2))
86             sigma = np.sqrt(np.log(variances[ward] / mean_stay[ward]**2 + 1)
87                 )
88             departures = np.random.poisson(ward_occupancy[ward] / lognorm.
89                 rvs(sigma, scale=np.exp(mean), size=1))
90             ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)
91
92     prob_all_beds_occupied = {ward: total_occupied_on_arrival[ward] / (
93         total_admissions[ward] + total_occupied_on_arrival[ward])
94         for ward in adjusted_capacities}
95     expected_admissions = {ward: total_admissions[ward] / days for ward in
96         adjusted_capacities}
97     expected_relocations = {ward: total_relocations[ward] / days for ward in
98         adjusted_capacities}
99
100     return total_admissions, total_relocations, total_losses, ward_occupancy
101         , prob_all_beds_occupied, expected_admissions, expected_relocations
102
103 # Set variances for log-normal distribution
```

```
96 variances_1 = {ward: 2 / (mean_stay[ward] ** 2) for ward in wards}
97 variances_2 = {ward: 3 / (mean_stay[ward] ** 2) for ward in wards}
98 variances_3 = {ward: 4 / (mean_stay[ward] ** 2) for ward in wards}
99
100 # Define optimal bed capacity for Ward F (previously found to be 27)
101 optimal_bed_capacity = 27
102
103 # Calculate adjusted capacities
104 adjusted_capacities = reallocate_beds(initial_capacities, urgency_points,
105                                     optimal_bed_capacity)
106
107 # Update relocation probabilities
108 relocation_probs_updated = update_relocation_probs(relocation_probs.copy())
109
110 # Run simulations with different variances
111 results_variances_1 = simulate_hospital_with_lognorm(365,
112                                     adjusted_capacities, relocation_probs_updated, variances_1)
113 results_variances_2 = simulate_hospital_with_lognorm(365,
114                                     adjusted_capacities, relocation_probs_updated, variances_2)
115 results_variances_3 = simulate_hospital_with_lognorm(365,
116                                     adjusted_capacities, relocation_probs_updated, variances_3)
117
118 # Define a function to format the results
119 def format_results(title, results):
120     total_admissions, total_relocations, total_losses, final_occupancy,
121     prob_all_beds_occupied, expected_admissions, expected_relocations =
122     results
123
124     print(f"\n{title}\n")
125     print(f"{'Ward':<5} {'Admissions':>12} {'Relocations':>12} {'Losses':>8}
126           {'Occupancy':>10} {'Prob Full':>10} {'Exp Admissions':>15} {'Exp
127           Relocations':>17}")
128     print("-" * 90)
129     for ward in wards:
130         occupancy = final_occupancy[ward][0] if isinstance(final_occupancy[
131             ward], np.ndarray) else final_occupancy[ward]
132         print(f"{ward:<5} {total_admissions[ward]:>12} {total_relocations[
133             ward]:>12} {total_losses[ward]:>8} {occupancy:>10} {
134             prob_all_beds_occupied[ward]:>10.3f} {expected_admissions[ward
135             ]:>15.3f} {expected_relocations[ward]:>17.3f}")
136
137 # Display results for each variance
138 format_results("Results with variance 2/Îž", results_variances_1)
139 format_results("Results with variance 3/Îž", results_variances_2)
```

```
128 format_results("Results with variance 4/ÎžÂ", results_variances_3)
```

## Sensitivity Analysis 2

```
1 import numpy as np
2 import pandas as pd
3 from scipy.stats import lognorm
4
5
6 import random
7 np.random.seed(42)
8
9 # Parameters
10 wards = ['A', 'B', 'C', 'D', 'E', 'F']
11 arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
12     13.0}
13 mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
14 initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20, 'F': 27}
15 urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
16 relocation_probs = {
17     'A': [0.00, 0.05, 0.10, 0.05, 0.80],
18     'B': [0.20, 0.00, 0.50, 0.15, 0.15],
19     'C': [0.30, 0.20, 0.00, 0.20, 0.30],
20     'D': [0.35, 0.30, 0.05, 0.00, 0.30],
21     'E': [0.20, 0.10, 0.60, 0.10, 0.00]
22 }
23
24 def update_relocation_probs(relocation_probs):
25     for ward in relocation_probs:
26         relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
27     relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
28     return relocation_probs
29
30 def simulate_hospital_with_lognorm(days, adjusted_capacities,
31     relocation_probs, variances):
32     ward_occupancy = {ward: 0 for ward in adjusted_capacities}
33     total_admissions = {ward: 0 for ward in adjusted_capacities}
34     total_relocations = {ward: 0 for ward in adjusted_capacities}
35     total_losses = {ward: 0 for ward in adjusted_capacities}
36     total_occupied_on_arrival = {ward: 0 for ward in adjusted_capacities}
37
38     for day in range(days):
```

```
37     for ward in adjusted_capacities:
38         arrivals = np.random.poisson(arrival_rates[ward])
39         for _ in range(arrivals):
40             if ward_occupancy[ward] < adjusted_capacities[ward]:
41                 ward_occupancy[ward] += 1
42                 total_admissions[ward] += 1
43             else:
44                 total_occupied_on_arrival[ward] += 1
45                 relocated = False
46                 for j, prob in enumerate(relocation_probs[ward]):
47                     if np.random.rand() < prob:
48                         alt_ward = list(adjusted_capacities.keys())[j]
49                         if ward_occupancy[alt_ward] <
                           adjusted_capacities[alt_ward]:
50                             ward_occupancy[alt_ward] += 1
51                             total_relocations[alt_ward] += 1
52                             relocated = True
53                             break
54                 if not relocated:
55                     total_losses[ward] += 1
56
57         mean = np.log(mean_stay[ward]**2 / np.sqrt(variances[ward] +
                           mean_stay[ward]**2))
58         sigma = np.sqrt(np.log(variances[ward] / mean_stay[ward]**2 + 1)
                           )
59         departures = np.random.poisson(ward_occupancy[ward] / lognorm.
                           rvs(sigma, scale=np.exp(mean), size=1))
60         ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)
61
62     prob_all_beds_occupied = {ward: total_occupied_on_arrival[ward] / (
                           total_admissions[ward] + total_occupied_on_arrival[ward])
63                               for ward in adjusted_capacities}
64     expected_admissions = {ward: total_admissions[ward] / days for ward in
                           adjusted_capacities}
65     expected_relocations = {ward: total_relocations[ward] / days for ward in
                           adjusted_capacities}
66
67     return total_admissions, total_relocations, total_losses, ward_occupancy
        , prob_all_beds_occupied, expected_admissions, expected_relocations
68
69 def format_results(title, results):
70     total_admissions, total_relocations, total_losses, final_occupancy,
        prob_all_beds_occupied, expected_admissions, expected_relocations =
        results
```

```
71
72     print(f"\n{title}\n")
73     print(f"{'Ward':<5} {'Admissions':>12} {'Relocations':>12} {'Losses':>8}
       {'Occupancy':>10} {'Prob Full':>10} {'Exp Admissions':>15} {'Exp
       Relocations':>17}")
74     print("-" * 90)
75     for ward in wards:
76         occupancy = final_occupancy[ward][0] if isinstance(final_occupancy[
            ward], np.ndarray) else final_occupancy[ward]
77         print(f"{'ward':<5} {'total_admissions[ward]:>12} {'total_relocations[
            ward]:>12} {'total_losses[ward]:>8} {'occupancy:>10} {'
            prob_all_beds_occupied[ward]:>10.3f} {'expected_admissions[ward
            ]:>15.3f} {'expected_relocations[ward]:>17.3f}")
78
79 # Different bed distribution scenarios
80 # Scenario 1: Increase beds in high-urgency wards (e.g., Ward D)
81 capacities_scenario1 = initial_capacities.copy()
82 capacities_scenario1['D'] += 10 # Increase beds in Ward D by 10
83 capacities_scenario1['A'] -= 3
84 capacities_scenario1['B'] -= 3
85 capacities_scenario1['C'] -= 2
86 capacities_scenario1['E'] -= 2
87 capacities_scenario1['F'] = 27 # Keep Ward F constant
88
89 # Scenario 2: Even distribution of beds
90 total_beds = sum(initial_capacities.values())
91 even_distribution = total_beds // len(wards)
92 capacities_scenario2 = {ward: even_distribution for ward in wards}
93
94 # Scenario 3: Increase beds in high-arrival wards (e.g., Wards A and F)
95 capacities_scenario3 = initial_capacities.copy()
96 capacities_scenario3['A'] += 10 # Increase beds in Ward A by 10
97 capacities_scenario3['F'] += 10 # Increase beds in Ward F by 10
98 capacities_scenario3['B'] -= 5
99 capacities_scenario3['C'] -= 5
100 capacities_scenario3['D'] -= 5
101 capacities_scenario3['E'] -= 5
102
103 # Set variances for log-normal distribution
104 variances = {ward: 2 / (mean_stay[ward] ** 2) for ward in wards}
105
106 # Run simulations for each scenario
107 relocation_probs_updated = update_relocation_probs(relocation_probs.copy())
```

```
108 results_scenario1 = simulate_hospital_with_lognorm(365, capacities_scenario1
    , relocation_probs_updated, variances)
109 results_scenario2 = simulate_hospital_with_lognorm(365, capacities_scenario2
    , relocation_probs_updated, variances)
110 results_scenario3 = simulate_hospital_with_lognorm(365, capacities_scenario3
    , relocation_probs_updated, variances)
111
112 # Display results for each scenario
113 format_results("Scenario 1: Increase beds in high-urgency wards",
    results_scenario1)
114 format_results("Scenario 2: Even distribution of beds", results_scenario2)
115 format_results("Scenario 3: Increase beds in high-arrival wards",
    results_scenario3)
```

## Sensitivity Analysis 3

```
1 import numpy as np
2 import pandas as pd
3 from scipy.stats import lognorm
4
5 # Set random seed for reproducibility
6 np.random.seed(42)
7
8 # Parameters
9 wards = ['A', 'B', 'C', 'D', 'E', 'F']
10 arrival_rates = {'A': 14.5, 'B': 11.0, 'C': 8.0, 'D': 6.5, 'E': 5.0, 'F':
    13.0}
11 mean_stay = {'A': 2.9, 'B': 4.0, 'C': 4.5, 'D': 1.4, 'E': 3.9, 'F': 2.2}
12 initial_capacities = {'A': 55, 'B': 40, 'C': 30, 'D': 20, 'E': 20}
13 urgency_points = {'A': 7, 'B': 5, 'C': 2, 'D': 10, 'E': 5}
14 relocation_probs = {
15     'A': [0.00, 0.05, 0.10, 0.05, 0.80],
16     'B': [0.20, 0.00, 0.50, 0.15, 0.15],
17     'C': [0.30, 0.20, 0.00, 0.20, 0.30],
18     'D': [0.35, 0.30, 0.05, 0.00, 0.30],
19     'E': [0.20, 0.10, 0.60, 0.10, 0.00]
20 }
21
22 def update_relocation_probs(relocation_probs):
23     for ward in relocation_probs:
24         relocation_probs[ward].append(0.0 if ward != 'F' else 1.0)
25     relocation_probs['F'] = [0.20, 0.20, 0.20, 0.20, 0.20, 0.0]
```

```
26     return relocation_probs
27
28 def reallocate_beds(total_beds, initial_capacities, urgency_points,
29                    bed_capacity_F):
30     remaining_beds = total_beds - bed_capacity_F
31     sorted_wards = sorted(initial_capacities.keys(), key=lambda x:
32                           urgency_points[x], reverse=True)
33     adjusted_capacities = {}
34
35     for ward in sorted_wards:
36         proportion = initial_capacities[ward] / sum(initial_capacities.
37                                                       values())
38         adjusted_capacities[ward] = int(proportion * remaining_beds)
39
40     current_total = sum(adjusted_capacities.values())
41     difference = remaining_beds - current_total
42
43     if difference > 0:
44         for ward in sorted_wards:
45             adjusted_capacities[ward] += 1
46             difference -= 1
47             if difference == 0:
48                 break
49
50     adjusted_capacities['F'] = bed_capacity_F
51     return adjusted_capacities
52
53 def simulate_hospital_with_lognorm(days, adjusted_capacities,
54                                    relocation_probs, variances):
55     ward_occupancy = {ward: 0 for ward in adjusted_capacities}
56     total_admissions = {ward: 0 for ward in adjusted_capacities}
57     total_relocations = {ward: 0 for ward in adjusted_capacities}
58     total_losses = {ward: 0 for ward in adjusted_capacities}
59     total_occupied_on_arrival = {ward: 0 for ward in adjusted_capacities}
60
61     for day in range(days):
62         for ward in adjusted_capacities:
63             arrivals = np.random.poisson(arrival_rates[ward])
64             for _ in range(arrivals):
65                 if ward_occupancy[ward] < adjusted_capacities[ward]:
66                     ward_occupancy[ward] += 1
67                     total_admissions[ward] += 1
68                 else:
69                     total_occupied_on_arrival[ward] += 1
```

```
66         relocated = False
67         for j, prob in enumerate(relocation_probs[ward]):
68             if np.random.rand() < prob:
69                 alt_ward = list(adjusted_capacities.keys())[j]
70                 if ward_occupancy[alt_ward] <
71                     adjusted_capacities[alt_ward]:
72                     ward_occupancy[alt_ward] += 1
73                     total_relocations[alt_ward] += 1
74                     relocated = True
75                     break
76             if not relocated:
77                 total_losses[ward] += 1
78
79         mean = np.log(mean_stay[ward]**2 / np.sqrt(variances[ward] +
80             mean_stay[ward]**2))
81         sigma = np.sqrt(np.log(variances[ward] / mean_stay[ward]**2 + 1)
82             )
83         departures = np.random.poisson(ward_occupancy[ward] / lognorm.
84             rvs(sigma, scale=np.exp(mean), size=1))
85         ward_occupancy[ward] = max(0, ward_occupancy[ward] - departures)
86
87     prob_all_beds_occupied = {ward: total_occupied_on_arrival[ward] / (
88         total_admissions[ward] + total_occupied_on_arrival[ward])
89         for ward in adjusted_capacities}
90     expected_admissions = {ward: total_admissions[ward] / days for ward in
91         adjusted_capacities}
92     expected_relocations = {ward: total_relocations[ward] / days for ward in
93         adjusted_capacities}
94
95     return total_admissions, total_relocations, total_losses, ward_occupancy
96         , prob_all_beds_occupied, expected_admissions, expected_relocations
97
98 def format_results(title, results):
99     total_admissions, total_relocations, total_losses, final_occupancy,
100     prob_all_beds_occupied, expected_admissions, expected_relocations =
101     results
102
103     print(f"\n{title}\n")
104     print(f"{'Ward':<5} {'Admissions':>12} {'Relocations':>12} {'Losses':>8}
105         {'Occupancy':>10} {'Prob Full':>10} {'Exp Admissions':>15} {'Exp
106             Relocations':>17}")
107     print("-" * 90)
108     for ward in wards:
```



```
97         occupancy = final_occupancy[ward][0] if isinstance(final_occupancy[
98             ward], np.ndarray) else final_occupancy[ward]
99     print(f"{ward:<5} {total_admissions[ward]:>12} {total_relocations[
100         ward]:>12} {total_losses[ward]:>8} {occupancy:>10} {
101             prob_all_beds_occupied[ward]:>10.3f} {expected_admissions[ward
102                 ]:>15.3f} {expected_relocations[ward]:>17.3f}")
103
104 # Define different total bed scenarios
105 total_beds_scenario1 = 170
106 total_beds_scenario2 = 180
107 total_beds_scenario3 = 150
108
109 # Set variances for log-normal distribution
110 variances = {ward: 2 / (mean_stay[ward] ** 2) for ward in wards}
111
112 # Calculate optimal bed capacity for Ward F (previously found to be 27)
113 optimal_bed_capacity_F = 27
114
115 # Calculate adjusted capacities for each scenario
116 adjusted_capacities_scenario1 = reallocate_beds(total_beds_scenario1,
117     initial_capacities, urgency_points, optimal_bed_capacity_F)
118 adjusted_capacities_scenario2 = reallocate_beds(total_beds_scenario2,
119     initial_capacities, urgency_points, optimal_bed_capacity_F)
120 adjusted_capacities_scenario3 = reallocate_beds(total_beds_scenario3,
121     initial_capacities, urgency_points, optimal_bed_capacity_F)
122
123 # Update relocation probabilities
124 relocation_probs_updated = update_relocation_probs(relocation_probs.copy())
125
126 # Run simulations for each scenario
127 results_scenario1 = simulate_hospital_with_lognorm(365,
128     adjusted_capacities_scenario1, relocation_probs_updated, variances)
129 results_scenario2 = simulate_hospital_with_lognorm(365,
130     adjusted_capacities_scenario2, relocation_probs_updated, variances)
131 results_scenario3 = simulate_hospital_with_lognorm(365,
132     adjusted_capacities_scenario3, relocation_probs_updated, variances)
133
134 # Display results for each scenario
135 format_results("Scenario 1: Total beds = 170", results_scenario1)
136 format_results("Scenario 2: Total beds = 180", results_scenario2)
137 format_results("Scenario 3: Total beds = 150", results_scenario3)
```