Case Assignment 3:

Hedging Risk: Case Analysis on GMS Investment Strategies

Group 9
Samadrita Roy Chowdhury, Anurag Reddy Katta, Dev Patel, Laxmi Lalitha Rachana Vepa

Case Background

- **Introduction**:
  - Kate Torelli, an analyst for Lion-Fund, identifies GMS stock as an attractive but risky investment.
  - Motivated by potential gold price increases and improved supply-demand conditions.
- **Challenges**:
  - GMS is highly leveraged, requiring a hedging strategy to mitigate downside risks.

Investment Scenarios

- **Stock Price Scenarios and Probabilities**:
  - 7 price scenarios ranging from $70 to $150, with given probabilities.
  - Visualize this using a **table or bar chart**.
- **Objective**: Minimize portfolio risk while optimizing returns on a $10 million investment.

Hedging Instruments

- **Put Options Overview**:
  - Strike prices and costs for three European-style options (A, B, C).
  - Example: Option A ($90 strike price) costs $2.20 per share.
- **Key Insight**: Put options provide value when GMS stock prices decline.

Decision Variables

- The decision variables will represent the amount of money that is invested in each security.
- Since there are three different put options (A, B, and C), the decision variables are:

x_{stock}: Amount of money invested in GMS stock.

x_{A}: Amount of money invested in Put Option A.

x_{B}: Amount of money invested in Put Option B.

x_{C}: Amount of money invested in Put Option C.

Constraints

- Budget Constraints: Total amount invested across all securities must be equal to the available investment capital, which is 10 million dollars.
- Non-negativity constraåints: The investment amount in each security cannot be negative.

```
#constraints
model.addConstr(sum(x[i] for i in range(4)) == total_investment, "TotalInvestment")
model.addConstrs((x[i] >= 0 for i in range(4)), "NonNegative")
```

Question 1 - GMS Stock Risk Analysis

- **Metrics for GMS Stock**:
  - Mean Return: 2.00%
  - Standard Deviation: 18.33%
- **Implication**: High variability indicates a significant risk for investors.

```python
initial_stock_price = 100
investment_amount = 10_000_000
num_shares = investment_amount / initial_stock_price

returns = []
probabilities = []
for scenario in scenarios.values():
    final_stock_price = scenario["stock_price"]
    stock_return = (final_stock_price - initial_stock_price) / initial_stock_price
    returns.append(stock_return)
    probabilities.append(scenario["prob"])

mean_return_gms = np.dot(returns, probabilities)
std_return_gms = np.sqrt(np.dot(probabilities, (np.array(returns) - mean_return_gms) ** 2))

{"Mean Return on GMS": mean_return_gms,
"Std Dev Return on GMS": std_return_gms}
```

```
{'Mean Return on GMS': 0.019999999999999997,
 'Std Dev Return on GMS': 0.18330302779823363}
```

Question 2 - Portfolio with Option A

- **Portfolio Analysis**:
  - Investment in GMS stock with Option A reduces overall risk.
  - **Metrics**:
    - Mean Return: 1.76%
    - Standard Deviation: 15.6%

```
option_a_strike = options["A"]["strike_price"]
option_a_cost = options["A"]["option_price"] * num_shares

returns_with_option_a = []
for scenario in scenarios.values():
    final_stock_price = scenario["stock_price"]
    option_payout = max(0, option_a_strike - final_stock_price) * num_shares
    total_return = ((final_stock_price * num_shares + option_payout - investment_amount - option_a_cost)
                    / (investment_amount + option_a_cost))
    returns_with_option_a.append(total_return)

mean_return_option_a = np.dot(returns_with_option_a, probabilities)
std_return_option_a = np.sqrt(np.dot(probabilities, (np.array(returns_with_option_a) - mean_return_option_a) ** 2))

{
"Mean Return with Option A": mean_return_option_a,
    "Std Dev Return with Option A": std_return_option_a
}
```

```
{'Mean Return with Option A': 0.01761252446183953,
 'Std Dev Return with Option A': 0.1559430278914797}
```

Question 3 - Optimal Portfolio

```
returns = []
for s in scenarios.values():
    stock_return = (s["stock_price"] / 100) - 1
    put_a_return = max(0, options["A"]["strike_price"] - s["stock_price"]) / options["A"]["option_price"] - 1
    put_b_return = max(0, options["B"]["strike_price"] - s["stock_price"]) / options["B"]["option_price"] - 1
    put_c_return = max(0, options["C"]["strike_price"] - s["stock_price"]) / options["C"]["option_price"] - 1
    weighted_return = sum(p * r for p, r in zip([x[i] for i in range(4)], [stock_return, put_a_return, put_b_return, put_c_returr
    returns.append(weighted_return)

mean_return = sum(s["prob"] * r for s, r in zip(scenarios.values(), returns))
for s, r in zip(scenarios.values(), returns):
    variance.add(s["prob"] * ((r - mean_return) ** 2))

model.setObjective(variance, GRB.MINIMIZE)
```

```
x[0]: 8491321.7623498
x[1]: 7.913785017357825e-15
x[2]: 6.222519840645219e-14
x[3]: 1508678.2376502007
```

```
import math
x = [84913.2, 0, 0, 15086.8] # xi to be the amount allocated to investment i in hundreds of dollars.

#returns
returns = []
for s in scenarios.values():
    stock_return = (s["stock_price"] / 100) - 1
    put_a_return = max(0, options["A"]["strike_price"] - s["stock_price"]) / options["A"]["option_price"] - 1
    put_b_return = max(0, options["B"]["strike_price"] - s["stock_price"]) / options["B"]["option_price"] - 1
    put_c_return = max(0, options["C"]["strike_price"] - s["stock_price"]) / options["C"]["option_price"] - 1
    weighted_return = sum(p * r for p, r in zip(x, [stock_return, put_a_return, put_b_return, put_c_return]))
    returns.append(weighted_return)

# mean return
mean_return = sum(s["prob"] * r for s, r in zip(scenarios.values(), returns))
print(f"Mean Return: {mean_return:.4f}")

# variance
variance = sum(s["prob"] * ((r - mean_return) ** 2) for s, r in zip(scenarios.values(), returns))
```

```
Mean Return: 1094.7920
Standard Deviation: 7951.7907
```

Question 4 - New Option Evaluation

```python
strike_price_new_option = 120

#fair price for the $120 option
fair_price_new_option = sum(
    scenario["prob"] * max(strike_price_new_option - scenario["stock_price"], 0)
    for scenario in scenarios.values()
)

print(f"Fair price of the $120 strike price option: {fair_price_new_option}")
```

```
Fair price of the $120 strike price option: 20.5
```

```python
options = {
    "A": {"strike_price": 90, "option_price": 2.20},
    "B": {"strike_price": 100, "option_price": 6.40},
    "C": {"strike_price": 110, "option_price": 12.50},
    "D": {"strike_price": 120, "option_price": 20.5},
}
```

```
x[0]: 7885860.182173415
x[1]: 1.1190036512590855e-16
x[2]: 1.2266620517437152e-15
x[3]: 7.636359972646247e-16
x[4]: 2114139.817826584
```

Conclusion

```python
import math
x = [78858.6, 0, 0, 0, 21141.38] # xi to be the amount allocated to investment i in hundreds of dollars.

#returns
returns = []
for s in scenarios.values():
    stock_return = (s["stock_price"] / 100) - 1
    put_a_return = max(0, options["A"]["strike_price"] - s["stock_price"]) / options["A"]["option_price"] - 1
    put_b_return = max(0, options["B"]["strike_price"] - s["stock_price"]) / options["B"]["option_price"] - 1
    put_c_return = max(0, options["C"]["strike_price"] - s["stock_price"]) / options["C"]["option_price"] - 1
    put_d_return = max(0, options["D"]["strike_price"] - s["stock_price"]) / options["D"]["option_price"] - 1
    weighted_return = sum(p * r for p, r in zip(x, [stock_return, put_a_return, put_b_return, put_c_return,put_d_return]))
    returns.append(weighted_return)

# mean return
mean_return = sum(s["prob"] * r for s, r in zip(scenarios.values(), returns))
print(f"Mean Return: {mean_return:.4f}")

# variance
variance = sum(s["prob"] * ((r - mean_return) ** 2) for s, r in zip(scenarios.values(), returns))
```

```
Mean Return: 1577.1720
Standard Deviation: 4645.5347
```