# Loss for Demand Predictions of an Online Store

You are tasked with forecasting the daily demand of an online retailer over a certain time period based on historical sales data. In addition to the sales data you are also provided with the stock (inventory) that was available for sale on each day. Therefore, the stock information provides an upper limit for the possible sales on each day. This small task is not concerned with the development of the model itself but only focuses on developing and implementing a tailor-suited loss function for this forecasting problem.

- Develop a loss function in PyTorch based on the code snippet below.
- Explain your choice and why it is a good fit for the problem at hand.
- Evaluate the loss function's performance with the provided dummy model. Adjust the code where needed to arrive at conclusions. Summarize and explain your findings.

```
import torch
import torch.nn as nn

# Define the input tensors for stocks and sales
n = 1000
torch.manual_seed(0)
stocks = torch.randint(0, 10, (n,))
demand = torch.poisson(torch.ones(n) * 2.0)
sales = torch.min(demand, stocks)

# Define the loss function
def forecast_loss(predictions, sales, stocks):
    # todo: implement the loss function here
    return loss

class MeanModel(nn.Module):
    def __init__(self):
        super(MeanModel, self).__init__()
        self.mean = nn.Parameter(torch.randn(1))

    def forward(self, n):
        return self.mean * torch.ones(n)


# Train the model and optimize using gradient descent
model = MeanModel()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

for i in range(100):
    optimizer.zero_grad()
    outputs = model(n)
    loss = forecast_loss(outputs, sales, stocks)
    loss.backward()
    optimizer.step()
    print("Epoch:", i, "Loss:", loss.item())
```