```python
import numpy as np
import matplotlib.pyplot as plt

def CalculatePolar(Cdo, CL, e, AR):
    CD = Cdo + (CL**2)/(np.pi * e * AR)
    return CD

def CalculateZeroLiftDrag(c, d, c_f, W_o, S):
    S_wet = (10**c) * (W_o**d)
    f = S_wet * c_f
    Cdo = f / S
    return Cdo

def BoxWingSpanEfficiency(e_ref, h, b):
    e_box = e_ref * (0.44 + 2.219 * (h/b)) / (0.44 + 0.9594 * (h/b))
    return e_box

# Given parameters
AR = 5.0
h = 10.36  # ft
b = 44  # ft
c = 1.0447  # Historical Value from Roskam
d = 0.5326  # Historical Value from Roskam
W_o = 8864
c_f = 0.0070
S = 387.2  # ft^2
e_ref = [0.825, 0.775, 0.725]  # Clean, Takeoff, Landing
delta_Cd = [0, 0.015, 0.065, 0.020]  # Clean, Takeoff flaps, Landing Flaps, Landing Gear

Cdo = CalculateZeroLiftDrag(c, d, c_f, W_o, S)
e_box = [BoxWingSpanEfficiency(e, h, b) for e in e_ref]

# CL max and min values for each configuration
CL_limits = {
    "Clean": (-1.6, 1.6),
    "Takeoff Flaps + Gear Up": (-1.73, 1.73),
    "Takeoff Flaps + Gear Down": (-1.73, 1.73),
    "Landing Flaps + Gear Up": (-2.0, 2.0),
    "Landing Flaps + Gear Down": (-2.0, 2.0)
}

# Configurations
takeoff_e = e_box[1]
landing_e = e_box[2]
configurations = {
    "Clean": (e_box[0], Cdo + delta_Cd[0], CL_limits["Clean"]),
    "Takeoff Flaps + Gear Up": (takeoff_e, Cdo + delta_Cd[1], CL_limits["Takeoff Flaps + Gear
Up"]),
    "Takeoff Flaps + Gear Down": (takeoff_e, Cdo + delta_Cd[1] + delta_Cd[3],
CL_limits["Takeoff Flaps + Gear Down"]),
    "Landing Flaps + Gear Up": (landing_e, Cdo + delta_Cd[2], CL_limits["Landing Flaps + Gear
Up"]),
    "Landing Flaps + Gear Down": (landing_e, Cdo + delta_Cd[2] + delta_Cd[3],
CL_limits["Landing Flaps + Gear Down"])
}

# Print span efficiency and zero-lift drag coefficient for each configuration
for label, (e, Cdo_mod, _) in configurations.items():
    print(f"{label}: Span Efficiency = {e:.4f}, Zero-Lift Drag Coefficient = {Cdo_mod:.4f}")

# Plotting drag polars
plt.figure(figsize=(8, 6))
for label, (e, Cdo_mod, (CL_min, CL_max)) in configurations.items():
    CL_range = np.linspace(CL_min, CL_max, 50)
    CD_values = [CalculatePolar(Cdo_mod, CL, e, AR) for CL in CL_range]
    plt.plot(CD_values, CL_range, label=label)
```

```
plt.xlabel("Drag Coefficient ($C_D$)")
plt.ylabel("Lift Coefficient ($C_L$)")
plt.title("Drag Polars for Box-Wing Aircraft Configurations")
plt.legend()
plt.grid()
plt.show()
```