1) CAPITAL BUDGETING PROBLEM

```
def capital_budgeting_dp(max_budget, num_plants, costs, returns):
    # Initialize the DP table
    dp = [[0 for _ in range(max_budget + 1)] for _ in range(num_plants + 1)]
    allocation = [[[] for _ in range(max_budget + 1)] for _ in range(num_plants + 1)]
    # Iterate over each plant
    for i in range(1, num_plants + 1):
        # Iterate over each budget from max_budget to 0
        for j in range(max_budget, -1, -1):
            # Iterate over each alternate
            for k in range(len(costs[i-1])):
                cost = costs[i-1][k]
                rtn = returns[i-1][k]
                if j >= cost:
                    if dp[i][j] < dp[i-1][j-cost] + rtn:
                        dp[i][j] = dp[i-1][j-cost] + rtn
                        allocation[i][j] = allocation[i-1][j-cost] + [(i, k)]
    return dp[num_plants][max_budget], allocation[num_plants][max_budget]
# Define the parameters
max_budget = 5
num_plants = 3
costs = [
    [0, 1, 2, 4],  # Costs for Plant 1
    [0, 2, 3, 4],  # Costs for Plant 2
    [0, 1, 2]      # Costs for Plant 3
]
returns = [
    [0, 15, 18, 28],  # Returns for Plant 1
    [0, 14, 18, 21],  # Returns for Plant 2
    [0, 3, 7]         # Returns for Plant 3
]
# Calculate the maximum return and allocations
max_return, allocations = capital_budgeting_dp(max_budget, num_plants, costs, returns)
print(f"The maximum return possible with a budget of {max_budget} crores is {max_return} crores.")
print("The stage-wise allocations are:")
for alloc in allocations:
    plant, alternate = alloc
    print(f"Plant -  {plant}, Alternate -  {alternate+1}, Cost: {costs[plant-1][alternate]}, Return: {returns[plant-1][alternate]}")
```

```
The maximum return possible with a budget of 5 crores is 36 crores.
The stage-wise allocations are:
Plant -  1, Alternate -  3, Cost: 2, Return: 18
Plant -  2, Alternate -  3, Cost: 3, Return: 18
Plant -  3, Alternate -  1, Cost: 0, Return: 0
```

2) PRODUCT ALLOCATION PROBLEM

```
def optimal_allocation(num_salesmen, num_zones, returns):
    # Initialize the DP table
    dp = [[0 for _ in range(num_salesmen + 1)] for _ in range(num_zones + 1)]
    allocation = [[[] for _ in range(num_salesmen + 1)] for _ in range(num_zones + 1)]
    # Iterate over each zone
    for i in range(1, num_zones + 1):
        # Iterate over each number of salesmen from 0 to num_salesmen
        for j in range(num_salesmen + 1):
            # Iterate over each possible number of salesmen in the current zone
            for k in range(min(j + 1, len(returns[i - 1]))):
                profit = returns[i-1][k]
                if dp[i][j] < dp[i-1][j-k] + profit:
                    dp[i][j] = dp[i-1][j-k] + profit
                    allocation[i][j] = allocation[i-1][j-k] + [(i, k)]
    return dp[num_zones][num_salesmen], allocation[num_zones][num_salesmen]
# Define the parameters
num_salesmen = 9
num_zones = 4
returns = [
    [45, 58, 70, 82, 93, 101, 108, 113, 118],  # sales for Zone 1
    [30, 45, 60, 70, 79, 90, 98, 105, 110],  # sales for Zone 2
    [35, 45, 52, 64, 72, 82, 93, 98, 100],  # sales for Zone 3
    [42, 54, 60, 70, 82, 95, 102, 110, 110]   # sales for Zone 4
]
# Calculate the maximum return and allocations
max_return, allocations = optimal_allocation(num_salesmen, num_zones, returns)
print(f"The maximum return possible with {num_salesmen} salesmen is {max_return}.")
print("The optimal allocation is as follows:")
```

```
for alloc in allocations:
    zone, salesmen = alloc
    print(f"Zone {zone}, Salesmen: {salesmen}")
```

```
The maximum return possible with 9 salesmen is 262.
The optimal allocation is as follows:
Zone 1, Salesmen: 4
Zone 2, Salesmen: 3
Zone 3, Salesmen: 1
Zone 4, Salesmen: 1
```