SOLVING REAL WORLD PROBLEMS USING COMPUTATIONAL THINKING

1.) Choose one real-world problem.

CHOSEN REAL WORLD PROBLEM: BUDGET MANAGEMENT IN A HOUSEHOLD

2.) Identify the problem:

(THIS IS A MONTHLY SALARY BASED)

The income on a certain household is enough for the needs like rent, groceries, electricity and water bills, but sometimes there are miscellaneous bills to pay like specific wants, debts, emergency bills, etc.

Since the income on the household is just enough what can you do to pay the miscellanous bills?

3.) Decomposition:

- Identify and categorize essential needs.
- Identify various types of miscellaneous bills.
- Assess total household income and determine its sufficiency for covering essential needs and miscellaneous bills.

4.) Abstraction:

- Prioritization of bills to ensure that essential needs are met first before addressing miscellaneous bills.
- Exclude or Deprioritize irrelevant bills that might complicate the budgeting process of the household.

KNAPSACK (SOURCE/INITIAL)

```
class Expense:
    def __init__(self, name, priority, cost):
        self.name = name
        self.priority = priority
        self.cost = cost

    def getPriority(self):
        return self.priority

    def getCost(self):
        return self.cost

    def getPriorityPerCost(self):
        return self.getPriority() / self.getCost()

    def __str__(self):
        return self.name + ': <' + str(self.priority) + ', ' + str(self.cost) + '>'

def createExpenses(names, priorities, costs):
    expenses = []
    for i in range(len(priorities)):
        expenses.append(Expense(names[i], priorities[i], costs[i]))
    return expenses
```

```
def knapsack(expenses, budget):
    if not expenses or budget <= 0:
        return [], 0

    first_expense = expenses[0]

    if first_expense.getCost() <= budget:
        with_first_expense, total_cost_with_first = knapsack(expenses[1:], budget - first_expense.getCost())
        with_first_expense.append(first_expense)
        total_cost_with_first += first_expense.getCost()
    else:
        with_first_expense, total_cost_with_first = [], 0

    without_first_expense, total_cost_without_first = knapsack(expenses[1:], budget)

    if total_cost_with_first > total_cost_without_first:
        return with_first_expense, total_cost_with_first
    else:
        return without_first_expense, total_cost_without_first

def doknapasacc(expenses, budget):
    selected, total_expense = knapsack(expenses, budget)
    print('\nTotal expenses of selected items =', total_expense)
    for expense in selected:
        print('  ', expense)

def Misc():
    miscellaneous = sum(cost) - salary
    print(f'Left budget for miscellaneous: {miscellaneous}')
```

USING DYNAMIC PROGRAMMING (TOP-DOWN APPROACH, MEMOIZATION, RECURSION)

```python
class Expense:
    def __init__(self, name, priority, cost):
        self.name = name
        self.priority = priority
        self.cost = cost

    def getPriority(self):
        return self.priority

    def getCost(self):
        return self.cost

    def getPriorityPerCost(self):
        return self.getPriority() / self.getCost()

    def __str__(self):
        return self.name + ': <' + str(self.priority) + ', ' + str(self.cost) + '>'

def createExpenses(names, priorities, costs):
    expenses = []
    for i in range(len(priorities)):
        expenses.append(Expense(names[i], priorities[i], costs[i]))
    return expenses

def knapsack_dynamic(expenses, budget, memo={}):
    if not expenses or budget <= 0:
        return [], 0

    if (len(expenses), budget) in memo:
        return memo[(len(expenses), budget)]

    first_expense = expenses[0]

    if first_expense.getCost() <= budget:
        with_first_expense, total_cost_with_first = knapsack_dynamic(expenses[1:], budget - first_expense.getCost(), memo)
        with_first_expense.append(first_expense)
        total_cost_with_first += first_expense.getCost()
    else:
        with_first_expense, total_cost_with_first = [], 0

    without_first_expense, total_cost_without_first = knapsack_dynamic(expenses[1:], budget, memo)

    if total_cost_with_first > total_cost_without_first:
        result = with_first_expense, total_cost_with_first
    else:
        result = without_first_expense, total_cost_without_first

    memo[(len(expenses), budget)] = result
    return result

def doknapasacc(expenses, budget):
    selected, total_expense = knapsack_dynamic(expenses, budget)
    print('\nTotal expenses of selected items =', total_expense)
    for expense in selected:
        print('  ', expense)

def Misc():
    miscellaneous = salary - sum(cost)
    print(f'Left budget for miscellaneous: {miscellaneous}')
    if miscellaneous < 0:
      print(f'I recommend you to cut back on some non essential spendings to satisfy your miscellaneous need.')

expenses = []
priority = []
cost = []

while True:
    exp = str(input("Enter your expense (leave empty when done): "))
    if not exp:
        break
    expenses.append(str(exp))
    print(f"\nBased on your expense, how important is {exp} in a rating out of 10?")
    while True:
        pri = input("Enter priority: ")
        if pri.isdigit():
            pri = float(pri)
            if pri <= 10:
                priority.append(pri)
                break
            else:
                print("Please enter a number equal or lower than 10.")
        else:
            print("Please enter a valid number for priority.")

    while True:
        cos = input("How much does it cost? (monthly): ")
        try:
            cost.append(float(cos))
            break
        except ValueError:
            print("Please enter a valid float for cost.")

    print()

print("\nYour expenses:")
for i in range(len(expenses)):
    print(f"  - {expenses[i]} (Priority: {priority[i]}, Cost: {cost[i]})")

expensess = createExpenses(expenses, priority, cost)

```

```
print()

salary = int(input('Enter Salary amount (monthly): '))
doknapasacc(expensess, salary)

print()

Misc()
```

```
Enter your expense (leave empty when done): RENT

Based on your expense, how important is RENT in a rating out of 10?
Enter priority: 10
How much does it cost? (monthly): 8790

Enter your expense (leave empty when done): WATER BILL

Based on your expense, how important is WATER BILL in a rating out of 10?
Enter priority: 7
How much does it cost? (monthly): 2345

Enter your expense (leave empty when done): ELECTRICITY BILL

Based on your expense, how important is ELECTRICITY BILL in a rating out of 10?
Enter priority: 7
How much does it cost? (monthly): 3456

Enter your expense (leave empty when done): GROCERIES

Based on your expense, how important is GROCERIES in a rating out of 10?
Enter priority: 8
How much does it cost? (monthly): 5670

Enter your expense (leave empty when done): NEW TIRES

Based on your expense, how important is NEW TIRES in a rating out of 10?
Enter priority: 5
How much does it cost? (monthly): 5000

Enter your expense (leave empty when done): BROKEN DOOR

Based on your expense, how important is BROKEN DOOR in a rating out of 10?
Enter priority: 4
Please enter a valid number for priority.
Enter priority: 4
How much does it cost? (monthly): 700

Enter your expense (leave empty when done): LEAK IN THE PIPES

Based on your expense, how important is LEAK IN THE PIPES in a rating out of 10?
Enter priority: 2300
Please enter a number equal or lower than 10.
Enter priority: 4
How much does it cost? (monthly): 2300

Enter your expense (leave empty when done): BROKEN DISHWASHER

Based on your expense, how important is BROKEN DISHWASHER in a rating out of 10?
Enter priority: 3
How much does it cost? (monthly): 2400

Enter your expense (leave empty when done): GTA VI

Based on your expense, how important is GTA VI in a rating out of 10?
Enter priority: 6
How much does it cost? (monthly): 2500
```