CASE STUDY 1: SOLVING REAL WORLD PROBLEMS USING COMPUTATIONAL THINKING

REAL WORLD PROBLEM:

budget management in a household

the monthly income on certain households is enough for the needs like rent, groceries, electricity and water bills, etc. but sometimes there are miscellaneous bills to pay for like specific wants, debts, emergency bills, maintenance, travel costs, etc.

CPE22S3-CPE311

TEAM 3

Belocora, John Rome

Calamba, Liam Francis

Montojo, Lance

+ Code + Text

KNAPSACK GRIDDY (SOURCE/INITIAL/TESTING/INSTANCES)

```
#testing area source code
#instances ↓↓↓
class Expense:
    def __init__(self, exp, value, priority):
        self.exp = exp
        self.value = value
        self.priority = priority
    def get_value(self):
        return self.value
    def get_priority(self):
        return self.priority
    def get_payfirst(self):
        return self.get_value() / self.get_priority()
    def __str__(self):
        return f"{self.exp}: <{self.value}, {self.priority}>"

def expense_list(exps, values, priority):
    menu = [Expense(exp, value, priority) for exp, value, priority in zip(exps, values, priority)]
    return menu
def greedy(items, max_value, key_function):
    items_copy = sorted(items, key=key_function, reverse=True)
    result = []
    total_value, total_cost = 0.0, 0.0
    for item in items_copy:
        if total_value + item.get_value() <= max_value:
            result.append(item)
            total_value += item.get_value()
            total_cost += item.get_priority()
    return result, total_value
def test_greedy(items, constraint, key_function):
    taken, val = greedy(items, constraint, key_function)
    print(f"Total value of expenses taken = {val}")
    for item in taken:
        print(f" {item}")
def test_greedys(Expenses, max_value):
    print(f"by value to allocate up to {max_value} PhP")
    test_greedy(Expenses, max_value, Expense.get_value)
    print(f"\nby priority to allocate up to {max_value} PhP")
    test_greedy(Expenses, max_value, lambda x: 1 / x.get_priority())
    print(f"\nYou should probably pay these first based on your salary \nwhich is: {max_value} PhP (montly)")
    test_greedy(Expenses, max_value, Expense.get_payfirst)

exps = ["RENT", "WATER BILL", "ELECTRICITY", "GROCERIES", "NEW TIRES", "BROKEN DOOR", "LEAK IN THE PIPES", "BROKEN DISWHWASHER"]
values = [8790, 2345, 3456, 5670, 5000, 700, 2300, 2400]
priority = [1, 3, 3, 2, 5, 6, 6, 7]
Expenses = expense_list(exps, values, priority)
test_greedys(Expenses, 27000)
```

```
by value to allocate up to 27000 PhP
Total value of expenses taken = 26016.0
 RENT: <8790, 1>
 GROCERIES: <5670, 2>
 NEW TIRES: <5000, 5>
 ELECTRICITY: <3456, 3>
 BROKEN DISWHWASHER: <2400, 7>
 BROKEN DOOR: <700, 6>

by priority to allocate up to 27000 PhP
Total value of expenses taken = 25961.0
 RENT: <8790, 1>
 GROCERIES: <5670, 2>
 WATER BILL: <2345, 3>
 ELECTRICITY: <3456, 3>
 NEW TIRES: <5000, 5>
 BROKEN DOOR: <700, 6>

You should probably pay these first based on your salary
which is: 27000 PhP (montly)
Total value of expenses taken = 25961.0
 RENT: <8790, 1>
```

```
 GROCERIES: <5670, 2>
 ELECTRICITY: <3456, 3>
 NEW TIRES: <5000, 5>
 WATER BILL: <2345, 3>
 BROKEN DOOR: <700, 6>
```

```
test_greedys(Expenses, 15000) #income is 15k
```

```
by value to allocate up to 15000 PhP
Total value of expenses taken = 14460.0
 RENT: <8790, 1>
 GROCERIES: <5670, 2>

by priority to allocate up to 15000 PhP
Total value of expenses taken = 14460.0
 RENT: <8790, 1>
 GROCERIES: <5670, 2>

You should probably pay these first based on your salary
which is: 15000 PhP (montly)
Total value of expenses taken = 14460.0
 RENT: <8790, 1>
 GROCERIES: <5670, 2>
```

```
test_greedys(Expenses, 25000) #income is 25k
```

```
by value to allocate up to 25000 PhP
Total value of expenses taken = 23616.0
 RENT: <8790, 1>
 GROCERIES: <5670, 2>
 NEW TIRES: <5000, 5>
 ELECTRICITY: <3456, 3>
 BROKEN DOOR: <700, 6>

by priority to allocate up to 25000 PhP
Total value of expenses taken = 23261.0
 RENT: <8790, 1>
 GROCERIES: <5670, 2>
 WATER BILL: <2345, 3>
 ELECTRICITY: <3456, 3>
 BROKEN DOOR: <700, 6>
 LEAK IN THE PIPES: <2300, 6>

You should probably pay these first based on your salary
which is: 25000 PhP (montly)
Total value of expenses taken = 23616.0
 RENT: <8790, 1>
 GROCERIES: <5670, 2>
 ELECTRICITY: <3456, 3>
 NEW TIRES: <5000, 5>
 BROKEN DOOR: <700, 6>
```

USING GREEDY ALGO USING DYNAMIC PROGRAMMING (BOTTOM UP, ITERATIVE)

```
#final code/ MAIN CODE
"""
ZPHRWWA
Jumpyyy
GRed
"""
class Expense:
    def __init__(self, exp, value, priority):
        self.exp = exp
        self.value = value
        self.priority = priority

    def get_value(self):
        return self.value
    def get_priority(self):
        return self.priority
    def get_payfirst(self):

        return self.get_value() / self.get_priority()
    def __str__(self):
        return f"{self.exp}: <{self.value}, {self.priority}>"

def expense_list(exps, values, priority):
    menu = [Expense(exp, value, priority) for exp, value, priority in zip(exps, values, priority)]
    return menu

def greedy(items, max_value, key_function):
    items_copy = sorted(items, key=key_function, reverse=True)
    result = []
    total_value, total_cost = 0.0, 0.0
    for item in items_copy:
        if total_value + item.get_value() <= max_value:
            result.append(item)
            total_value += item.get_value()
            total_cost += item.get_priority()
    return result, total_value

def test_greedy(items, constraint, key_function):
    taken, val = greedy(items, constraint, key_function)
    print(f"Total value of expenses taken = {val}")
    for item in taken:
        print(f" {item}")

```

```
def test_greedys(Expenses, max_value):
    print(f"\nby value to allocate up to {max_value} PhP")
    test_greedy(Expenses, max_value, Expense.get_value)
    print(f"\nby priority to allocate up to {max_value} PhP")
    test_greedy(Expenses, max_value, lambda x: 1 / x.get_priority())
    print(f"\n You should probably pay these first based on your salary \n which is: {max_value} PhP (montly)")
    test_greedy(Expenses, max_value, Expense.get_payfirst)

def get_user_input(message):
    """Gets user input and returns it as a string"""
    while True:
        try:
            return input(message)
        except ValueError:
            print("Invalid input. Please enter a string.")

def create_expense(exp, value, priority):
    """Creates an Expense object with the given attributes"""
    return Expense(exp, value, priority)

def build_expense_list():
    """Prompts user for expense details and creates a list of Expense objects"""
    expenses = []
    while True:
        exp = get_user_input("Enter an expense (or 'q' to quit): ")
        if exp.lower() == 'q':
            break
        value = float(get_user_input("Enter expense value: "))
        priority = int(get_user_input("Enter expense priority (1 as top priority 10 as least priority ): "))
        expenses.append(create_expense(exp, value, priority))
    return expenses

def get_value_limit():
    """Gets user input for the value limit and validates it"""
    while True:
        try:
            limit = float(get_user_input("How much is your monthly budget: "))
            print()
            if limit > 0:
                return limit
            else:
                print("Invalid input. Budget must be positive.")
        except ValueError:
            print("Invalid input. Please enter a number.")

exps = []  # Empty list to store expense exps
values = []  # Empty list to store expense values
priority = []  # Empty list to store expense priorities

# Get user input for the priority limit
priority_limit = get_value_limit()

# Call the user input function to build the expense list
expenses = build_expense_list()

# Convert expense list to exps, values, and priority lists
for expense in expenses:
  exps.append(expense.exp)
  values.append(expense.value)
  priority.append(expense.priority)

test_greedys(expenses, priority_limit)
```

```
Enter expense value: 700
Enter expense priority (1 as top priority 10 as least priority ): 6
Enter an expense (or 'q' to quit): LEAK IN THE ROOF
Enter expense value: 2300
Enter expense priority (1 as top priority 10 as least priority ): 5
Enter an expense (or 'q' to quit): GROCERIES
Enter expense value: 5746
Enter expense priority (1 as top priority 10 as least priority ): 2
Enter an expense (or 'q' to quit): NEW TIRES
Enter expense value: 5000
Enter expense priority (1 as top priority 10 as least priority ): 6
Enter an expense (or 'q' to quit): NEW DESK
Enter expense value: 1500
Enter expense priority (1 as top priority 10 as least priority ): 4
Enter an expense (or 'q' to quit): SOLDERING TOOLS
Enter expense value: 700
Enter expense priority (1 as top priority 10 as least priority ): 7
Enter an expense (or 'q' to quit): WATER BILL
Enter expense value: 2437
Enter expense priority (1 as top priority 10 as least priority ): 3
Enter an expense (or 'q' to quit): ELECTRICITY BILL
Enter expense value: 2778
Enter expense priority (1 as top priority 10 as least priority ): 3
Enter an expense (or 'q' to quit): RENT
Enter expense value: 8777
Enter expense priority (1 as top priority 10 as least priority ): 1
Enter an expense (or 'q' to quit): Q

by value to allocate up to 30000.0 PhP
Total value of expenses taken = 29438.0
 RENT: <8777.0, 1>
 GROCERIES: <5746.0, 2>
 NEW TIRES: <5000.0, 6>
```

```
 RENT: <8777.0, 1>
 GROCERIES: <5746.0, 2>
 WATER BILL: <2437.0, 3>
 ELECTRICITY BILL: <2778.0, 3>
 NEW DESK: <1500.0, 4>
 LEAK IN THE ROOF: <2300.0, 5>
 BROKEN DOOR: <700.0, 6>
 NEW TIRES: <5000.0, 6>
 SOLDERING TOOLS: <700.0, 7>

 You should probably pay these first based on your salary
 which is: 30000.0 PhP (montly)
Total value of expenses taken = 29938.0
 RENT: <8777.0, 1>
 GROCERIES: <5746.0, 2>
 ELECTRICITY BILL: <2778.0, 3>
 NEW TIRES: <5000.0, 6>
 WATER BILL: <2437.0, 3>
 LEAK IN THE ROOF: <2300.0, 5>
```