LOGICAL PROGRAMMING ALGORITHM
STUDY CASE

Muhammad Brian Abdillah	(24091397052)
Achmad Hakim Aljumadi	(24091397099)
Arsya Vanessa	(24091397116)

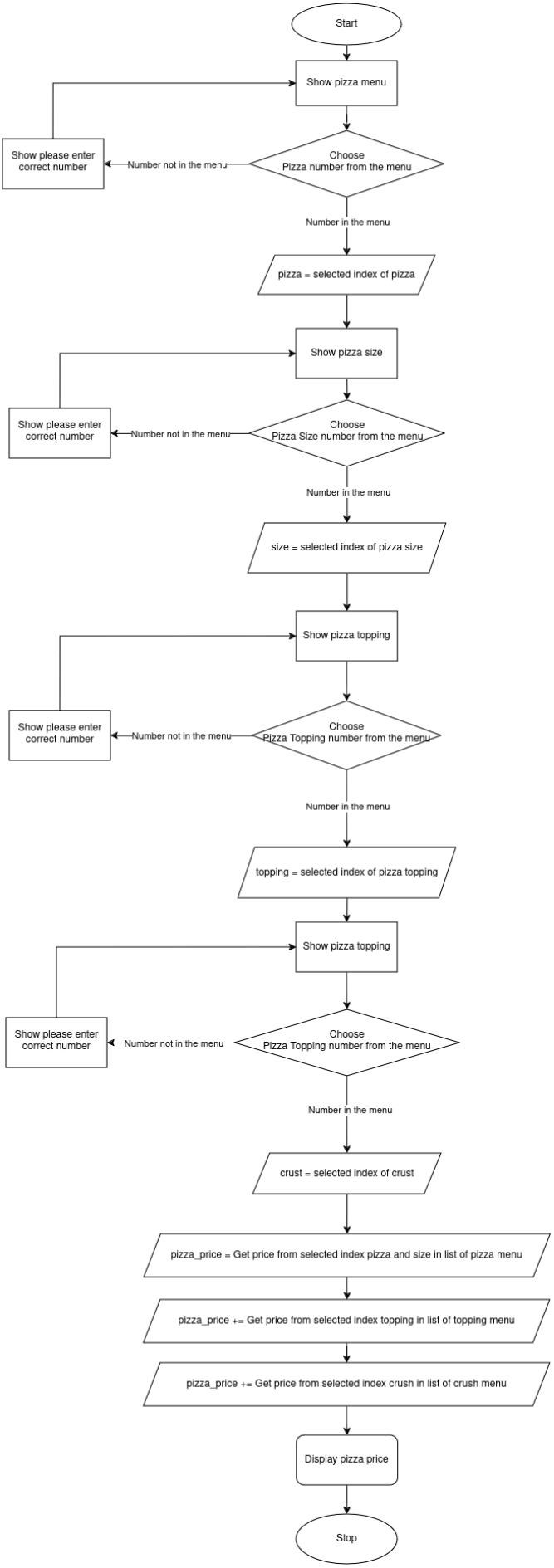
D4 Informatics Management Study Program

Faculty of Vocational

Surabaya State University

2024

FLOWCHART


Start
Show pizza menu
Choose
Pizza number from the menu
Number not in the menu
Show please enter
correct number
Number in the menu
pizza = selected index of pizza
Show pizza size
Choose
Pizza Size number from the menu
Number not in the menu
Show please enter
correct number
Number in the menu
size = selected index of pizza size
Show pizza topping
Choose
Pizza Topping number from the menu
Number not in the menu
Show please enter
correct number
Number in the menu
topping = selected index of pizza topping
Show pizza topping
Choose
Pizza Topping number from the menu
Number not in the menu
Show please enter
correct number
Number in the menu
crust = selected index of crust
pizza_price = Get price from selected index pizza and size in list of pizza menu
pizza_price += Get price from selected index topping in list of topping menu
pizza_price += Get price from selected index crush in list of crush menu
Display pizza price
Stop

EXPLANATION:

This flowchart represents a pizza ordering system, where the user selects a pizza from the menu, chooses its size, toppings, crust, and then calculates the total price based on the selections. Below is a step-by-step explanation of the flowchart:

1. Start: The system begins by showing the pizza menu to the user.
2. Show Pizza Menu: A list of available pizzas is displayed for the user to choose from.
3. Choose Pizza Number from the Menu: The user is prompted to select a pizza number from the menu. Decision:
    a. If the number is in the menu: The system proceeds with the selected pizza, storing its index (variable pizza = selected index of pizza).
    b. If the number is not in the menu: The system asks the user to enter a correct number and goes back to the pizza menu.
4. Show Pizza Size: Once the pizza is selected, the system shows the available sizes for the chosen pizza.
5. Choose Pizza Size Number from the Menu: The user selects a size for the pizza. Decision
    a. If the size number is in the menu: The system proceeds with the selected size, storing its index (variable size = selected index of pizza size).
    b. If the size number is not in the menu: The system asks the user to enter a correct number and goes back to the size menu.
6. Show Pizza Topping: After selecting the size, the system shows the available toppings.
7. Choose Pizza Topping Number from the Menu: The user selects a topping. Decision:
    a. If the topping number is in the menu: The system proceeds with the selected topping, storing its index (variable topping = selected index of pizza topping).
    b. If the topping number is not in the menu: The system asks the user to enter a correct number and goes back to the topping menu.
8. Show Pizza Crust: After selecting toppings, the system displays the available crust options.
9. Choose Pizza Crust Number from the Menu: The user selects a crust type. Decision:
    a. If the crust number is in the menu: The system proceeds with the selected crust, storing its index (variable crust = selected index of crust).
    b. If the crust number is not in the menu: The system asks the user to enter a correct number and goes back to the crust menu.
10. Calculate Pizza Price: The system calculates the pizza price based on the user's choices:
    a. pizza_price = Get price from selected index pizza and size in list of pizza menu
    b. pizza_price += Get price from selected index topping in list of topping menu
    c. pizza_price += Get price from selected index crust in list of crust menu
11. Display Pizza Price: The total pizza price is displayed to the user based on their selections.
12. Stop: The process concludes, and the system completes the order.

The system allows the user to navigate through pizza, size, topping, and crust options, ensuring the correct menu item is selected at each step before calculating and displaying the final price.

CODE SNIPPETS & EXPLANATION

1. Menu Overview

```
# Map of Menu
'''
No. Toppings        small   regular large
1.  Margherita      63.750  85.000  106.250
2.  Pepperoni       71.250  95.000  118.750
3.  Meat Lovers     67.500  90.000  112.500
4.  Super Supreme   71.250  95.000  118.750
5.  Cheeseburger    67.500  90.000  112.500
6.  Double Cheese   63.500  86.000  107.500

Additional
Extra Meat of Choice    +13.000
Extra Cheese            +13.000
Extra Basil             +6.000

Crust Price
Thin Crust              +4.000
Thick Crust             +7.000
Pan Crust               +4.000
Cheese Burst Crust      +13.000
Stuffed Crust           +10.000
'''
```

The provided code snippet serves as a foundational reference for a pizza ordering system. By transitioning to more structured data formats and implementing functions for interaction and calculations, the system can be made more efficient and user-friendly.

2. Pizza Menu

```
# Pizza menu
pizza_menu = {
    '1': {'name': 'Margherita', 'small' : 63.750, 'regular': 85.000, 'large': 106.250},
    '2': {'name': 'Pepperoni', 'small' : 71.250, 'regular': 95.000, 'large': 118.750},
    '3': {'name': 'Meat Lovers', 'small' : 67.500, 'regular': 90.000, 'large': 112.500},
    '4': {'name': 'Super Supreme', 'small' : 71.250, 'regular': 95.000, 'large': 118.750},
    '5': {'name': 'Cheeseburger', 'small' : 67.500, 'regular': 90.000, 'large': 112.500},
    '6': {'name': 'Double Cheese', 'small' : 63.500, 'regular': 86.000, 'large': 107.500}
}
```

This code defines a dictionary, *pizza_menu*, which stores information about different types of pizzas available in a menu.

The dictionary keys *('1', '2', etc.)* represent different pizza options by a unique identifier.

Each pizza entry contains another dictionary as its value, with the following fields:

- *'name'*: The name of the pizza.
- *'small', 'regular',* and *'large'*: The prices of the pizza for different sizes (small, regular, large).

Output:

```
Pizza Menu
1. Margherita
2. Pepperoni
3. Meat Lovers
4. Super Supreme
5. Cheeseburger
6. Double Cheese

Choose your pizza: 6
```

In terminal the user choose number 6 (Double Cheese) after the user choose what pizza they wanted, the program will save the option and add the price, in this case it count as a regular size pizza Rp, 86.000,-

3. Pizza Size Entries

```
34  # Pizza size
35  # price calculated in percentage
36  pizza_size = {
37      '1': {'name': 'Small', 'price': -25},
38      '2': {'name': 'Regular', 'price': 0},
39      '3': {'name': 'Large', 'price': 25}
40  }
```

Similar to the pizza menu, the dictionary keys *('1', '2', '3')* represent different pizza sizes by a unique identifier. Each size entry contains two fields:
'name': The size of the pizza (Small, Regular, Large).
'price': A percentage adjustment to be applied to the base price, likely used to modify the price based on size.

- *'1'*: Represents a Small pizza with a -25% price adjustment, meaning a small pizza costs 25% less than the base price.
- *'2'*: Represents a Regular pizza with a 0% price adjustment, meaning it costs the same as the base price (no change).
- *'3'*: Represents a Large pizza with a +25% price adjustment, meaning a large pizza costs 25% more than the base price.

Output:

```
Pizza Size
1. Small: -25%
2. Regular: 0%
3. Large: 25%

Choose your size: 1
```

In terminal the user choose number 1 (Small) after the user choose what pizza size they want, the program will save the option and adjust the price, originally it was Rp, 86.000,- then program will reduce its price to -25% of the original price. after that it'll be Rp, 63.500,-

4. **Topping Entry**

```
42  # Pizza topping
43  pizza_topping = {
44      '1': {'name': 'None', 'price': 0},
45      '2': {'name': 'Extra Meat of Choice', 'price': 13},
46      '3': {'name': 'Extra Cheese', 'price': 13},
47      '4': {'name': 'Extra Basil', 'price': 6}
48  }
```

The dictionary keys *('1', '2', '3', '4')* represent unique identifiers for each pizza topping.
Each topping entry contains two fields:
'name': The name of the topping
'price': The price to be added for that topping.
Here's a breakdown of all the toppings:

- *'1'*: None (no topping) with a price of 0 (no extra cost).
- *'2'*: Extra Meat of Choice with a price of 13 (adds 13 units to the base price).
- *'3'*: Extra Cheese with a price of 13 (adds 13 units to the base price).
- *'4'*: Extra Basil with a price of 6 (adds 6 units to the base price).

Imagine we have a base pizza price of 100 units, and the user selects *Extra Meat* of Choice (key *'2'*): The topping adds 13 units to the base price. 100+13=113
The *pizza_topping* dictionary allows you to store different toppings with their prices in a structured way. The keys are used to easily retrieve the name and price of a topping.
The price is then added to the base pizza price when calculating the total cost of the pizza order.
Output:

```
1. None: 0k
2. Extra Meat of Choice: 13k
3. Extra Cheese: 13k
4. Extra Basil: 6k

Choose your topping: 4
```

In terminal the user choose number 4 (Extra Basil) after the user choose what pizza topping they want, the program will save the option and adjust the price, from Rp, 63.500,- the program will add Rp, 6.000,- so it'll be Rp, 69.500,-

5. **Crust Entry**

```
50  # Pizza crust
51  pizza_crust = {
52      '1': {'name': 'Normal Crust', 'price': 0},
53      '2': {'name': 'Thin Crust', 'price': 4},
54      '3': {'name': 'Thick Crust', 'price': 7},
55      '4': {'name': 'Pan Crust', 'price': 4},
56      '5': {'name': 'Cheese Burst Crust', 'price': 13},
57      '6': {'name': 'Stuffed Crust', 'price': 10}
58  }
```

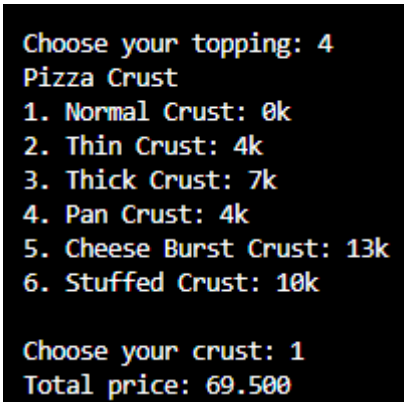
The dictionary keys *('1', '2', '3', etc.)* are unique identifiers for each type of pizza crust.
Each key maps to another dictionary containing two fields:
'name': The name of the pizza crust (e.g., "Normal Crust").

'price': The additional price that will be added to the base pizza price if this crust is selected. Here's all the crusts and their corresponding prices:

- *'1'*: Normal Crust with a price of 0 (no additional cost).
- *'2'*: Thin Crust with a price of 4 (adds 4 units to the base price).
- *'3'*: Thick Crust with a price of 7 (adds 7 units to the base price).
- *'4'*: Pan Crust with a price of 4 (adds 4 units to the base price).
- *'5'*: Cheese Burst Crust with a price of 13 (adds 13 units to the base price).
- *'6'*: Stuffed Crust with a price of 10 (adds 10 units to the base price).

This code helps manage pizza crust selection in a pizza-ordering system by associating each crust with an additional price that adjusts the final total. By selecting the crust type, the program can add the corresponding price to the base pizza price to get the final cost.

Output:



In terminal the user choose number 1 (Normal Crust) after the user choose what pizza crust they want, the program will save the option and adjust the price, since normal crust cost Rp, 0,- so it's still be Rp, 69.500,-

6. Range Input

```
60  def range_input(msg, first, end):
61      """
62          Prompt the user to input an integer within a specified range and ensure the user provides a valid input.
63      """
64      while True:
65          try:
66              value = int(input(msg))
67              if value < first or value > end:
68                  raise ValueError
69              return str(value)
70          except ValueError:
71              print(f"Please input number between {first} and {end}")
```

This code defines a function called *range_input*, which is used to prompt the user for an integer input within a specified range. It ensures that the input is valid (within the given range) and handles invalid inputs appropriately.

Continuously prompt the user for an integer until they enter a valid one. The valid integer must fall between *first* and *end* (inclusive). If the user enters an invalid value (non-integer or an out-of-range number), the function informs the user and prompts them to try again.

- Infinite Loop (while True): The function runs an infinite loop to repeatedly ask for input until a valid number is entered.
- Input and Integer Conversion: The function uses *input(msg)* to prompt the user for input with the message *msg*. It attempts to convert the input to an integer using *int()*.

If the input is not a valid integer, it raises a *ValueError*, and the program moves to the exception handler.

- Range Check: After successfully converting the input to an integer, the function checks if the number falls outside the specified range *(first to end)*. If the number is outside this range, a *ValueError* is manually raised, which leads to the exception handling block. If the input is valid, the function converts the number to a string and returns it. If a *ValueError* is raised (either because the input is not an integer or it's out of range), the exception handler prints a message instructing the user to enter a valid number within the specified range.

7. Pizza Menu

```
73  def print_menu():
74      """
75          Show the pizza menu.
76      """
77      print("Pizza Menu")
78      for key, value in pizza_menu.items():
79          print(f"{key}. {value['name']}")
80      print()
```

This code defines a function called *print_menu()* that displays a list of pizza options from a predefined *pizza_menu* dictionary.

- Function Definition: This defines a function named *print_menu()* which does not take any arguments. It is intended to display the pizza menu when called.
- Docstring: The docstring briefly describes the purpose of the function: "Show the pizza menu." This is useful for developers to understand the function's role when reading the code.
- Menu Header: "Pizza Menu" indicates that the subsequent output is related to the available pizza options.
- Loop through the *pizza_menu*: This line iterates over the *pizza_menu* dictionary. The dictionary is expected to store details of the pizzas, where *key:* Represents the unique identifier for each pizza (like *'1', '2'*, etc.). And *value*: A dictionary containing details about each pizza
- Print Each Pizza Option: The function prints a formatted string showing *key*: The unique identifier for the pizza. *value['name']*: The name of the pizza.
- The menu items are printed in a numbered format, making it easy for the user to select a pizza by its key.
- Final Print Statement: This empty *print()* statement is used to add a blank line after the menu has been printed, improving the readability of the output.

8. Pizza Size

```
82  def print_size():
83      """
84          Show the pizza size.
85      """
86      print("Pizza Size")
87      for key, value in pizza_size.items():
88          print(f"{key}. {value['name']}: {value['price']}%")
89      print()
```

This code defines a function called *print_size()* that displays a list of available pizza sizes and their associated price adjustments based on different pizza sizes and their relative price differences (as percentages).

For each pizza size in the menu, the function prints:

key: The unique identifier for the size ; *value['name']*: The name of the size ; *value['price']*: The price adjustment percentage for that size. This value is printed with a percentage symbol *%*, indicating that the price is increased or decreased by a certain percentage.

- Small size: The base price is reduced by 25%.
- Regular size: No change to the base price (0% adjustment).
- Large size: The base price is increased by 25%.

The *value['price']* field in each entry represents the percentage change in the base price of the pizza depending on the size. This allows the pizza's final price to vary based on the size selected by the customer. For example, a smaller pizza may be cheaper (negative percentage), while a larger one may cost more (positive percentage).

9. Pizza Toppings

```
91  def print_topping():
92      """
93          Show the pizza topping.
94      """
95      print("Pizza Topping")
96      for key, value in pizza_topping.items():
97          print(f"{key}. {value['name']}: {value['price']}k")
98      print()
```

This code defines a function *print_topping()* that displays pizza toppings and their prices in a formatted way.

For each topping, the function prints:

key: The unique identifier for the topping ; *value['name']:* The name of the topping ; *value['price']*: The price of that topping, displayed with a suffix 'k' (indicating that the price is measured in thousands, or whatever 'k' represents in this context).

The items are displayed in a numbered format for easy selection.

The *value['price']* field represents the cost of each topping, and displaying it with the suffix *'k'* suggests that the values are either in thousands or represent some other unit (like currency). This format is beneficial for users to understand the additional costs associated with each topping.

10. Pizza Crust

```
100  def print_crust():
101      """
102          Show the pizza crust.
103      """
104      print("Pizza Crust")
105      for key, value in pizza_crust.items():
106          print(f"{key}. {value['name']}: {value['price']}k")
107      print()
```

The *print_crust()* function is intended to present the available pizza crust options to the user, including each crust's unique identifier and its price.

The *value['price']* field represents the cost of each crust, and displaying it with the suffix *'k'* suggests that the values are either in thousands or represent some other unit (like currency). This format helps users understand the additional costs associated with each crust type. The *print_crust()* function effectively displays available pizza crust options and their prices from a predefined dictionary (*pizza_crust*). By looping through the dictionary, it prints each crust type along with its price in a clear and user-friendly format.

11. Final Order

```
109  if __name__ == "__main__":
110      # Show the menu
111      print_menu()
112      pizza = range_input("Choose your pizza: ", 1, len(pizza_menu))
113      print_size()
114      size = range_input("Choose your size: ", 1, len(pizza_size))
115      print_topping()
116      topping = range_input("Choose your topping: ", 1, len(pizza_topping))
117      print_crust()
118      crust = range_input("Choose your crust: ", 1, len(pizza_crust))
119
120      # Calculate the price
121      pizza_price = pizza_menu[pizza][pizza_size[size]['name'].lower()]
122      pizza_price += pizza_topping[topping]['price']
123      pizza_price += pizza_crust[crust]['price']
124
125      # Show the total price
126      print(f"Total price: {format(pizza_price, '.3f')}")
```

This code block is the main execution part of a pizza ordering program. It interacts with the user to select a pizza, size, topping, and crust type, calculates the total price based on those selections, and displays the total price

- The *if __name__ == "__main__"*: block ensures that the code runs only when the script is executed directly, not when it is imported as a module. The code relies on several previously defined functions (*print_menu(), print_size(), print_topping(), print_crust()*) and dictionaries (*pizza_menu, pizza_size, pizza_topping, pizza_crust*).
- *pizza = range_input("Choose your pizza: ", 1, len(pizza_menu))* prompts the user to select a pizza by calling the *range_input()* function, which ensures that the user inputs a valid integer between *1* and the number of available pizzas

- *size = range_input("Choose your size: ", 1, len(pizza_size))* Similar to the pizza selection, this line prompts the user to select a size, ensuring valid input, and stores the result in the variable *size*.
- *topping = range_input("Choose your topping: ", 1, len(pizza_topping))* This prompts the user to select a topping, ensuring valid input, and stores the result in the variable *topping*.
- *crust = range_input("Choose your crust: ", 1, len(pizza_crust))* This prompts the user to select a crust, ensuring valid input, and stores the result in the variable *crust.*
- *Calculate the price* : This block calculates the total price of the pizza by combining the price of the selected pizza, size, topping, and crust. The topping and crust prices are added to the *pizza_price* variable.
- *print(f"Total price: {format(pizza_price, '.3f')}")* This line prints the total price of the pizza formatted to three decimal places.

CODE PREVIEW

```
# Map of Menu
'''
No. Toppings        small   regular large
1.  Margherita      63.750  85.000  106.250
2.  Pepperoni       71.250  95.000  118.750
3.  Meat Lovers     67.500  90.000  112.500
4.  Super Supreme   71.250  95.000  118.750
5.  Cheeseburger    67.500  90.000  112.500
6.  Double Cheese   63.500  86.000  107.500

Additional
Extra Meat of Choice    +13.000
Extra Cheese            +13.000
Extra Basil             +6.000

Crust Price
Thin Crust              +4.000
Thick Crust             +7.000
Pan Crust               +4.000
Cheese Burst Crust      +13.000
Stuffed Crust           +10.000

# Pizza menu
pizza_menu = {
    '1': {'name': 'Margherita', 'small' : 63.750, 'regular': 85.000, 'large': 106.250},
    '2': {'name': 'Pepperoni', 'small' : 71.250, 'regular': 95.000, 'large': 118.750},
    '3': {'name': 'Meat Lovers', 'small' : 67.500, 'regular': 90.000, 'large': 112.500},
    '4': {'name': 'Super Supreme', 'small' : 71.250, 'regular': 95.000, 'large': 118.750},
    '5': {'name': 'Cheeseburger', 'small' : 67.500, 'regular': 90.000, 'large': 112.500},
    '6': {'name': 'Double Cheese', 'small' : 63.500, 'regular': 86.000, 'large': 107.500}
}

# Pizza size
# price calculated in percentage
pizza_size = {
    '1': {'name': 'Small', 'price': -25},
    '2': {'name': 'Regular', 'price': 0},
    '3': {'name': 'Large', 'price': 25}
}

```

```python
# Pizza topping
pizza_topping = {
    '1': {'name': 'None', 'price': 0},
    '2': {'name': 'Extra Meat of Choice', 'price': 13},
    '3': {'name': 'Extra Cheese', 'price': 13},
    '4': {'name': 'Extra Basil', 'price': 6}
}

# Pizza crust
pizza_crust = {
    '1': {'name': 'Normal Crust', 'price': 0},
    '2': {'name': 'Thin Crust', 'price': 4},
    '3': {'name': 'Thick Crust', 'price': 7},
    '4': {'name': 'Pan Crust', 'price': 4},
    '5': {'name': 'Cheese Burst Crust', 'price': 13},
    '6': {'name': 'Stuffed Crust', 'price': 10}
}
def range_input(msg, first, end):
    """
        Prompt the user to input an integer within a specified range and ensure the user provides a valid input.
    """
    while True:
        try:
            value = int(input(msg))
            if value < first or value > end:
                raise ValueError
            return str(value)
        except ValueError:
            print(f"Please input number between {first} and {end}")

def print_menu():
    """
        Show the pizza menu.
    """
    print("Pizza Menu")
    for key, value in pizza_menu.items():
        print(f"{key}. {value['name']}")
    print()

def print_size():
    """
        Show the pizza size.
    """
    print("Pizza Size")
    for key, value in pizza_size.items():
        print(f"{key}. {value['name']}: {value['price']}%")
    print()

def print_topping():
    """
        Show the pizza topping.
    """
    print("Pizza Topping")
    for key, value in pizza_topping.items():
        print(f"{key}. {value['name']}: {value['price']}k")
    print()

def print_crust():
    """
        Show the pizza crust.
    """
    print("Pizza Crust")
    for key, value in pizza_crust.items():
        print(f"{key}. {value['name']}: {value['price']}k")
    print()

if __name__ == "__main__":
    # Show the menu
    print_menu()
    pizza = range_input("Choose your pizza: ", 1, len(pizza_menu))
    print_size()
    size = range_input("Choose your size: ", 1, len(pizza_size))
    print_topping()
    topping = range_input("Choose your topping: ", 1, len(pizza_topping))
    print_crust()
    crust = range_input("Choose your crust: ", 1, len(pizza_crust))

    # Calculate the price
    pizza_price = pizza_menu[pizza][pizza_size[size]['name'].lower()]
    pizza_price += pizza_topping[topping]['price']
    pizza_price += pizza_crust[crust]['price']

    # Show the total price
    print(f"Total price: {format(pizza_price, '.3f')}")

```

TERMINAL PREVIEW

```
Pizza Menu
1. Margherita
2. Pepperoni
3. Meat Lovers
4. Super Supreme
5. Cheeseburger
6. Double Cheese

Choose your pizza: 6
Pizza Size
1. Small: -25%
2. Regular: 0%
3. Large: 25%

Choose your size: 1
Pizza Topping
1. None: 0k
2. Extra Meat of Choice: 13k
3. Extra Cheese: 13k
4. Extra Basil: 6k

Choose your topping: 4
Pizza Crust
1. Normal Crust: 0k
2. Thin Crust: 4k
3. Thick Crust: 7k
4. Pan Crust: 4k
5. Cheese Burst Crust: 13k
6. Stuffed Crust: 10k

4. Extra Basil: 6k

Choose your topping: 4
Pizza Crust
1. Normal Crust: 0k
2. Thin Crust: 4k
3. Thick Crust: 7k
4. Pan Crust: 4k
5. Cheese Burst Crust: 13k
6. Stuffed Crust: 10k

Pizza Crust
1. Normal Crust: 0k
2. Thin Crust: 4k
3. Thick Crust: 7k
4. Pan Crust: 4k
5. Cheese Burst Crust: 13k
6. Stuffed Crust: 10k

2. Thin Crust: 4k
3. Thick Crust: 7k
4. Pan Crust: 4k
5. Cheese Burst Crust: 13k
6. Stuffed Crust: 10k

Choose your crust: 1
Total price: 69.500
PS C:\Users\Administrator>


Total price: 69.500
PS C:\Users\Administrator>
```

CONCLUSION :
The code is a pizza ordering system that allows users to select a pizza, choose its size, add toppings, and pick a crust type. It calculates the final price based on these choices.

Pizza Menu: Offers six pizzas with three size options (small, regular, large), each with different prices.
Customizable Options:

- Size: Adjusts the price by ±25% for small or large sizes.
- Toppings: Offers extra toppings like meat, cheese, or basil, with fixed additional costs.

Crust: Users can choose from various crust types, each with a different price (e.g., Stuffed Crust, Cheese Burst).
Price Calculation: The final price starts with the base pizza price (regular size), adjusts for size, and adds costs for extra toppings and crust.
User Interaction: The system prompts users for inputs, validates them, and calculates the total cost.

In essence, this system provides an interactive, customizable pizza ordering experience, dynamically adjusting the total price based on user selections.

LINK GITHUB : https://github.com/brianabdl/Pizza-Cashier-Algorithm