AS 91373 Practice Assessment

Algorithm

```
START
        regularPizza = 8.5
        gourmetPizza = 13.5
        pizzaMenu = {“Cheese” : regularPizza,
                “Pepperoni” : regularPizza,
                “Hawaiian” : regularPizza,
                “Ham & Cheese” : regularPizza,
                “BBQ Pork & Onion” : regularPizza,
                “Beef & Onion” : regularPizza,
                “Cheesy Garlic” : regularPizza,
                “Sweet & Sour Chicken” : gourmetPizza,
                “Mega Meatlovers” : gourmetPizza,
                “Garlic Prawn”: gourmetPizza,
                “BBQ Chicken & Rasher Bacon” : gourmetPizza,
                “Butter Chicken” : gourmetPizza,
                }
        dashedLine = “------------------------------------------------------------"
        DEFINE userInput(rangeMaximum, inputMessage, ifError, exceptError) AS:
                inputValue = “”
                WHILE inputValue IS NOT INTEGER OR NOT IN RANGE(1, rangeMaximum + 1) DO:
                        inputValue = INPUT(inputMessage)
                        TRY THE FOLLOWING:
                                inputValue = INTEGER(inputValue)
                                IF inputValue NOT IN RANGE(1, rangeMaximum + 1) DO:
                                        PRINT(ifError)
                        EXCEPT:
                                GO TO cancelOrder(inputValue)
                                PRINT(exceptError)
                RETURN inputValue
        DEFINE cancelOrder(checkInput) AS:
```

```
        IF LOWERCASE checkInput IS EQUAL TO "x" DO:
            GO TO restartOrExit
DEFINE restartOrExit AS:
    PRINT("[NEW-LINE]Would you like to enter another order, or exit the program?[NEW-LINE][NEW-LINE]1) Enter another order[NEW-LINE]2) Exit program")
    restartInput = userInput(2, "[NEW-LINE]Enter your choice: ", "[NEW-LINE]Your choice must be either 1 or 2.", "[NEW-LINE]Your choice must be an integer that is either 1 or 2.")
    IF restartInput IS EQUAL TO 1 DO:
        GO TO orderFunction
    ELSE IF restartInput IS EQUAL TO 2 DO:
        EXIT PROGRAM
DEFINE customerInformation(pickupOrDelivery) AS:
    customerAddress = NONE
    customerNumber = NONE
    orderCost = 0
    customerName = INPUT("[NEW-LINE]Input the customer's name: ")
    GO TO cancelOrder(customerName)
    IF pickupOrDelivery IS EQUAL TO 2 DO:
        INCREASE orderCost BY 3
        customerAddress = INPUT("[NEW-LINE]Input the customer's address: ")
        GO TO cancelOrder(customerAddress)
        WHILE NOT customerNumber IS INTEGER DO:
            customerNumber = INPUT("NEW-LINE]Input the customer's phone number: ")
            TRY THE FOLLOWING:
                customerNumber = INTEGER(customerNumber)
            EXCEPT:
                GO TO cancelOrder(customerNumber)
                PRINT("[NEW-LINE]The customer's phone number must not contain any letters, symbols or spaces.")
    RETURN customerName, customerAddress, customerNumber, orderCost
DEFINE orderFunction AS:
    customerOrdered = []
    PRINT("[NEW-LINE]Is the order for pick-up or delivery?[NEW-LINE][NEW-LINE]1) Pick-up[NEW-LINE]2) Delivery")
```

```
        pickupOrDelivery = userInput(2, "[NEW-LINE]Enter your selection: ", "[NEW-LINE] Your selection must be either 1 or 2.", "[NEW-LINE]Your selection must be an integer that is either 1 or 2.")

        customerName, customerAddress, customerNumber, orderCost = customerInformation(pickupOrDelivery)

        PRINT("[NEW-LINE]How many pizzas are to be ordered?[NEW-LINE][NEW-LINE]1) 1 pizza[NEW-LINE]2) 2 pizzas[NEW-LINE]3) 3 pizzas[NEW-LINE]4) 4 pizzas[NEW-LINE]5) 5 pizzas")

        numberOfPizzas = userInput(5, "[NEW-LINE]Number of pizzas: ", "[NEW-LINE]The number of pizzas must be between 1 and 5.", "[NEW-LINE]The number of pizzas must be an integer between 1 and 5.")

        PRINT("[NEW-LINE]Which pizzas would the customer like to order?[NEW-LINE]")

        FOR menuNumber, pizzaName IN ENUMERATE(pizzaMenu, 1) DO:

            PRINT("[menuNumber]) [pizzaName]")

        FOR orderNumber IN RANGE(0, numberOfPizzas) DO:

            APPEND customerOrdered WITH userInput(LENGTH(pizzaMenu), "[NEW-LINE]Enter the corresponding number: ", "[NEW-LINE]Your input must be between 1 and [LENGTH(pizzaMenu)].", "[NEW-LINE]Your input must be an integer between 1 and [LENGTH(pizzaMenu)].")

        FOR orderedIndex IN RANGE(0, LENGTH(customerOrdered)) DO:

            FOR menuNumber, pizzaName IN ENUMERATE(pizzaMenu, 1) DO:

                IF customerOrdered[orderedIndex] IS EQUAL TO menuNumber DO:

                    customerOrdered[orderedIndex] = pizzaName

        PRINT("[NEW-LINE][dashedLine][NEW-LINE]")

        FOR pizzaNumber, pizzaName IN ENUMERATE(customerOrdered, 1) DO:

            PRINT("[pizzaNumber]) [pizzaName] – $[pizzaMenu[pizzaName]]")

            INCREASE orderCost by pizzaMenu[pizzaName]

        PRINT("[NEW-LINE]Total cost of order: $[orderCost]")

        PRINT("[NEW-LINE]Customer's name: [customerName]")

        IF pickupOrDelivery IS EQUAL TO 2 DO:

            PRINT("[NEW-LINE]Customer's address: [customerAddress]")

            PRINT("[NEW-LINE]Customer's phone number: [customerNumber]")

        PRINT("[NEW-LINE][dashedLine]")

        GO TO restartOrExit

    PRINT("If you would like to cancel the order at any time, enter 'x' (case insensitive).")

    GO TO orderFunction

END
```

Test Plan

Colour	What is being tested	rangeMaximum	inputValue	Test type	Expected result	Actual result	How it was fixed	Date tested
	Check if user's integer is valid and within specified range	2	1	Expected / Boundary	Passes through function, inputValue is returned successfully	InputValue is returned successfully		16APR19
			2					
			0	Boundary	Caught by IF statement with error message, WHILE loop triggers and input is re-done	'Your selection must be either 1 or 2.'		
			3					
			6	Invalid				
			ABCD		Caught by EXCEPT with error message, WHILE loop triggers and input is re-done	'Your selection must be an integer that is either 1 or 2.'		

Colour	What is being tested	rangeMaximum	inputValue	Test type	Expected result	Actual result	How it was fixed	Date tested
	Check if user's integer input is valid and within specified range	5	3	Expected	Passes through function, inputValue is returned successfully	InputValue is returned successfully		16APR19
			1	Expected / Boundary				
			5					
			0	Boundary	Caught by IF statement with error message, WHILE loop triggers and input is re-done	'The number of pizzas must be between 1 and 5.'		
			6					
			9	Invalid				
			ABCD		Caught by EXCEPT with error message, WHILE loop triggers and input is re-done	'The number of pizzas must be an integer between 1 and 5.'		

Colour	What is being tested	rangeMaximum	inputValue	Test type	Expected result	Actual result	How it was fixed	Date tested
	Check if user's integer input is valid and within specified range	LENGTH(pizzaMenu)	3	Expected	Passes through function, inputValue is returned successfully	InputValue is returned successfully		16APR19
			1	Expected / Boundary				
			LENGTH(pizzaMenu)					
			0	Boundary	Caught by IF statement with error message, WHILE loop triggers and input is re-done	'Your input must be between 1 and 12.'		
			LENGTH(pizzaMenu) + 1					
			LENGTH(pizzaMenu) + 5	Invalid				
			ABCD		Caught by EXCEPT with error message, WHILE loop triggers and input is re-done	'Your input must be an integer between 1 and 12.'		

Colour	What is being tested	checkInput	Test type	Expected result	Actual result	How it was fixed	Date tested
	Check if the user's input is equal to 'x' (or 'X' as case is insensitive)	x	Expected	Caught by IF statement, goes to restartOrExit	Goes to restartOrExit		16APR19
		X					
		ABCD		Passes through function and resumes with remainder of code from where it was called	Continues with code		
		John					
		30 Forrest Hill Rd					

Colour	What is being tested	restartInput	Test type	Expected result	Actual result	How it was fixed	Date tested
	Determine whether the user wishes to enter another order to exit the program	1	Expected	Caught by IF statement, goes to orderFunction	Goes to orderFunction		16APR19
		2		Caught by ELSE IF, exits the program	Program exits		

Colour	What is being tested	pickupOrDelivery	Test type	Expected result	Actual result	How it was fixed	Date tested
	Check if order is for delivery, and if so prompts for the additional required information	1	Expected	Passes through and returns customerName as the input remaining variables as their initial values (None)	Asks for customer's name then moves on		16APR19

Colour	What is being tested	pickupOrDelivery	Test type	Expected result	Actual result	How it was fixed	Date tested
	Check if order is for delivery, and if so prompts for the additional required information	2	Expected	Caught by IF statement, adds the $3 delivery charge, and prompts for additional information on customer (address & phone number)	Asks for customer's name, address and phone number then moves on		16APR19

Colour	What is being tested	customerNumber	Test type	Expected result	Actual result	How it was fixed	Date tested
	Check if user's integer input is valid	0211234567	Expected	Passes through and returns variables as respective inputs	Successful input, code continues		16APR19
		021 123 4567	Invalid	Caught by EXCEPT with error message, WHILE loop triggers and input is re-done	'The customer's phone number must not contain any letters, symbols or spaces.'		
		021-123-4567					
		ABCD					

Colour	What is being tested	ordered Index	customerOrdered[orderedIndex]	menu Number	Test type	Expected result	Actual result	How it was fixed	Date tested
	Check to see if the number is being correctly changed to the corresponding pizza name	2	4	4	Expected	Replaces the value at customerOrdered[2] from 4 to the pizza name at pizzaMenu[4]	Value changed '4' > 'Ham & Cheese'		16APR19
		4	1	1			Value changed '1' > 'Cheese'		
		2	4	2		Continue to next iteration in the FOR loop	Iterations continue		

Colour	What is being tested	pickupOrDelivery	Test type	Expected result	Actual result	How it was fixed	Date tested
	Check if order is for delivery, and if so print the additional required information	1	Expected	Passes through and goes to restartOrExit	'1) Cheese – $8.5 Total cost of order: $8.5 Customer's name: James'		16APR19
		2		Caught by IF statement, prints the additional required information	'1) Cheese – $8.5 Total cost of order: $11.5 Customer's name: James Customer's address: 30 Forrest Hill Road Customer's phone number: 211234567'		

Program Code (new version)

```
import sys

def user_input(rangeMaximum, promptWord):
    inputValue = ""

    while inputValue not in range(MINIMUM_INPUT, rangeMaximum + RANGE_OFFSET):
        try:
            inputValue = int(input("\nEnter {}: ".format(promptWord)))

            if inputValue not in range(MINIMUM_INPUT, rangeMaximum + RANGE_OFFSET):
                print("\n{} must be between {} and {}.".format(promptWord.capitalize(), MINIMUM_INPUT,
rangeMaximum))

        except:
            print("\n{} must be an integer between {} and {}.".format(promptWord.capitalize(),
MINIMUM_INPUT, rangeMaximum))

    return inputValue

def numbered_output(dataStructure):
    for number, item in enumerate(dataStructure, ENUMERATE_START):
        print("{}) {}".format(number, item))

def restart_or_exit():
    ACTION_OPTIONS = ("Enter another order", "Exit program")
    print("\nWould you like to enter another order, or exit the program?\n")
    numbered_output(ACTION_OPTIONS)
    restartInput = user_input(len(ACTION_OPTIONS), "your choice")

    if ACTION_OPTIONS[restartInput - INDEX_OFFSET] == "Exit program":
        sys.exit()

def customer_information(pickupOrDelivery):
    customerName = ""
    customerAddress = None
    customerNumber = None
    orderCost = 0

    while not customerName.isalpha():
        customerName = input("\nInput the customer's first name: ")

        if not customerName.isalpha():
            print("\nThe customer's first name must only contain letters A-Z.")

    if pickupOrDelivery == "Delivery":
        orderCost += DELIVERY_COST

        customerAddress = input("\nInput the customer's address: ")
```

```
        while not isinstance(customerNumber, int)
            try:
                customerNumber = int(input("\nInput the customer's phone number: "))

            except:
                print("\nThe customer's phone number must not contain any letters, symbols or spaces.")

    return customerName, customerAddress, customerNumber, orderCost

DASHED_LINE = 72
REGULAR_PIZZA = 8.5
GOURMET_PIZZA = 13.5
DELIVERY_COST = 3
MINIMUM_INPUT = 1
RANGE_OFFSET = 1
INDEX_OFFSET = 1
INDEX_START = 0
ENUMERATE_START = 1
PIZZA_MENU = {"Cheese":REGULAR_PIZZA,
              "Pepperoni":REGULAR_PIZZA,
              "Hawaiian":REGULAR_PIZZA,
              "Ham & Cheese":REGULAR_PIZZA,
              "BBQ Pork & Onion":REGULAR_PIZZA,
              "Beef & Onion":REGULAR_PIZZA,
              "Cheesy Garlic":REGULAR_PIZZA,
              "Sweet & Sour Chicken":GOURMET_PIZZA,
              "Mega Meatlovers":GOURMET_PIZZA,
              "Garlic Prawn":GOURMET_PIZZA,
              "BBQ Chicken & Rasher Bacon":GOURMET_PIZZA,
              "Butter Chicken":GOURMET_PIZZA,
              }

print("Welcome to Dream Pizzas' new order input system!")

while True:
    customerOrdered = []

    OBTAINING_OPTIONS = ("Pick-up", "Delivery")
    print("\nIs the order for pick-up or delivery?\n")
    numbered_output(OBTAINING_OPTIONS)
    pickupOrDelivery = user_input(len(OBTAINING_OPTIONS), "your selection")

    customerName, customerAddress, customerNumber, orderCost =
customer_information(OBTAINING_OPTIONS[pickupOrDelivery - INDEX_OFFSET])

    QUANTITY_OPTIONS = ("1 pizza", "2 pizzas", "3 pizzas", "4 pizzas", "5 pizzas")
    print("\nHow many pizzas are to be ordered?\n")
    numbered_output(QUANTITY_OPTIONS)
    numberOfPizzas = user_input(len(QUANTITY_OPTIONS), "the number of pizzas")

    print("\nWhich pizzas would the customer like to order?\n")
```

```
numbered_output(PIZZA_MENU)

for orderNumber in range(INDEX_START, numberOfPizzas):
    customerOrdered.append(user_input(len(PIZZA_MENU), "the corresponding number"))

for orderedIndex in range(INDEX_START, len(customerOrdered)):
    for menuNumber, pizzaName in enumerate(PIZZA_MENU, ENUMERATE_START):
        if customerOrdered[orderedIndex] == menuNumber:
            customerOrdered[orderedIndex] = pizzaName

CONFIRM_OPTIONS = ("Confirm order", "Cancel order")
print("\nWould you like to confirm the order, or cancel the order?\n")
numbered_output(CONFIRM_OPTIONS)
confirmOrCancel = user_input(len(CONFIRM_OPTIONS), "your choice")

if CONFIRM_OPTIONS[confirmOrCancel - INDEX_OFFSET] == "Cancel order":
    restart_or_exit()
    continue

print("\n{}\n".format("-" * DASHED_LINE))

for pizzaNumber, pizzaName in enumerate(customerOrdered, ENUMERATE_START):
    print("{}) {} – ${}0".format(pizzaNumber, pizzaName, PIZZA_MENU[pizzaName]))
    orderCost += PIZZA_MENU[pizzaName]

print("\nTotal cost of order: ${}0".format(orderCost))
print("\nCustomer's name: {}".format(customerName))

if OBTAINING_OPTIONS[pickupOrDelivery - INDEX_OFFSET] == "Delivery":
    print("\nCustomer's address: {}".format(customerAddress))
    print("\nCustomer's phone number: {}".format(customerNumber))

print("\n{}".format("-" * DASHED_LINE))

restart_or_exit()
```