AS 91372 Practice Assessment

Purpose

The purpose of the brief is to create a program for Dream Pizzas, which allows them to enter customer details, pizza(s) ordered and pick-up or delivery requirements and displays the delivery details, itemised list of pizzas ordered, and the total cost of the order.

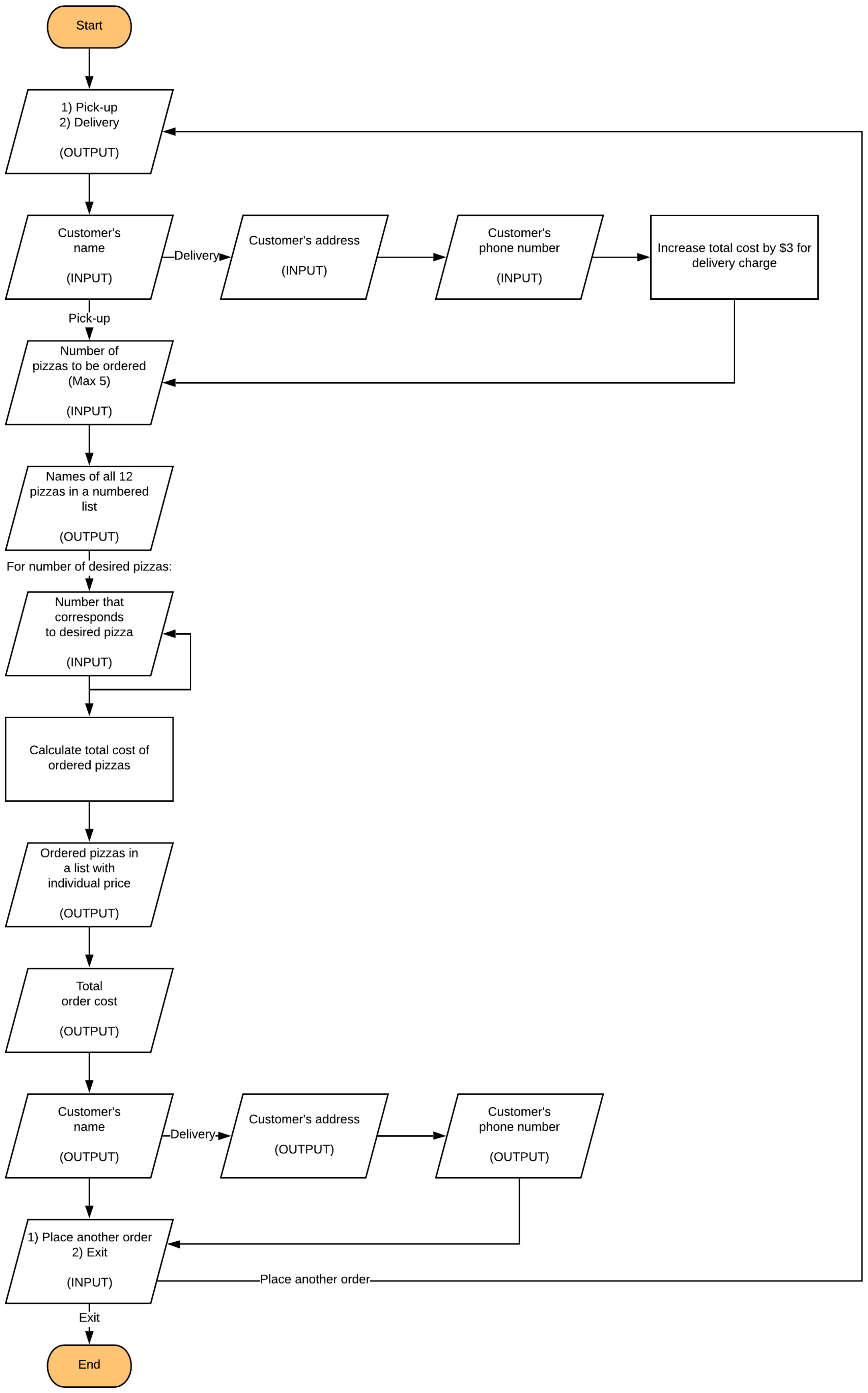
Target Audience

The target audience of this program are the employees at Dream Pizzas who receive the pizza orders via telephone.

Target Language

This program will be created using Python 3.4.4.

Flow Chart


Start
1) Pick-up
2) Delivery
(OUTPUT)
Customer's name
(INPUT)
Delivery
Customer's address
(INPUT)
Customer's phone number
(INPUT)
Increase total cost by $3 for delivery charge
Pick-up
Number of pizzas to be ordered (Max 5)
(INPUT)
Names of all 12 pizzas in a numbered list
(OUTPUT)
For number of desired pizzas:
Number that corresponds to desired pizza
(INPUT)
Calculate total cost of ordered pizzas
Ordered pizzas in a list with individual price
(OUTPUT)
Total order cost
(OUTPUT)
Customer's name
(OUTPUT)
Delivery
Customer's address
(OUTPUT)
Customer's phone number
(OUTPUT)
1) Place another order
2) Exit
(INPUT)
Place another order
Exit
End

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
```

Data Dictionary

Data item	Data type	Scope	Data location	Variable name	Initial value
Cost of a regular pizza	Float	Global		regularPizza	8.5
Cost of a gourmet pizza				gourmetPizza	13.5
All pizzas on the menu	Dictionary			pizzaMenu	See top of algorithm
Constant for outputted dashed line	String			dashedLine	
Function for integer inputs with validity checking	Function			userInput	
The maximum value allowed for integer inputs	Argument	Local	userInput	rangeMaximum	
The message that prompts input				inputMessage	
The error message to be printed if the integer is out of the allowed range				ifError	
The error message to be printed if input is not an integer				exceptError	
The user's integer input	Integer			inputValue	
Function that checks if the user's input is 'x'	Function	Global		cancelOrder	
The input value to be checked	Argument	Local	cancelOrder	checkInput	
Function that determines whether to exit or input another order	Function	Global		restartOrExit	
User's input for whether to input another order or exit the program	Integer	Local	restartOrExit	restartInput	
Function for input of customer's information	Function	Global		customerInformation	

Data item	Data type	Scope	Data location	Variable name	Initial value
Input for whether order is pick-up or delivery	Argument	Local	customerInformation	pickupOrDelivery	
Customer's name	String	Local	customerInformation	customerName	
Customer's address	String	Local	customerInformation	customerAddress	None
Customer's phone number	Integer	Local	customerInformation	customerNumber	None
Total cost of the order	Integer	Local	customerInformation	orderCost	0
Main function of the program	Function	Global		orderFunction	
List of customer's ordered pizzas	List	Local	orderFunction	customerOrdered	[]
Input for whether order is pick-up or delivery	Integer	Local	orderFunction	pickupOrDelivery	
Customer's name	String	Local	orderFunction	customerName	
Customer's address	String	Local	orderFunction	customerAddress	
Customer's phone number	Integer	Local	orderFunction	customerNumber	
Total cost of the order	Integer	Local	orderFunction	orderCost	
Number of pizzas the customer is to order	Integer	Local	orderFunction	numberOfPizzas	
Number which corresponds to each pizza on the menu	Argument	Local	orderFunction	menuNumber	
Name of each pizza on the menu	Argument	Local	orderFunction	pizzaName	
The number pizza currently being ordered, out of the total to be ordered	Argument	Local	orderFunction	orderNumber	
The index at which the customer's order is within customerOrdered	Argument	Local	orderFunction	orderedIndex	
Number which corresponds to customer's ordered pizza	Argument	Local	orderFunction	pizzaNumber	

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			
			2					
			0	Boundary	Caught by IF statement with error message, WHILE loop triggers and input is re-done			
			3					
			6	Invalid				
			ABCD		Caught by EXCEPT with error message, WHILE loop triggers and input is re-done			

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			
			1	Expected / Boundary				
			5					
			0	Boundary	Caught by IF statement with error message, WHILE loop triggers and input is re-done			
			6					
			9	Invalid				
			ABCD		Caught by EXCEPT with error message, WHILE loop triggers and input is re-done			

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			
			1	Expected / Boundary				
			LENGTH(pizzaMenu)					
			0	Boundary	Caught by IF statement with error message, WHILE loop triggers and input is re-done			
			LENGTH(pizzaMenu) + 1					
			LENGTH(pizzaMenu) + 5	Invalid				
			ABCD		Caught by EXCEPT with error message, WHILE loop triggers and input is re-done			

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			
		X					
		ABCD		Passes through function and resumes with remainder of code from where it was called			
		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			
		2		Caught by ELSE IF, exits the program			

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)			

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)			

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			
		021 123 4567	Invalid	Caught by EXCEPT with error message, WHILE loop triggers and input is re-done			
		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]			
		4	1	1					
		2	4	2		Continue to next iteration in the FOR loop			

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			
		2		Caught by IF statement, prints the additional required information			