Pre-release Material Development

Computer Science

March 7, 2019

This document covers what the question demands and how it came to be answered. The following is present in this document.

* A copy of the question
* Paraphrasing of the question
* Rough outlines of the expected program
* Test data and expected results
* Any problems encountered during development
* How any problems encountered were resolved
* Identifiers used in the final program.

Anuj Verma

Delhi Public School Bangalore East [IGCSE]

0478/22/PRE/F/M/19

# Question

# Requirements

A pizza ordering service allows customization. The options provided are shown below:

|  |  |  |
| --- | --- | --- |
| Property | Options | Rules |
| Size | Small, Medium, Large | Any **one** |
| Base | Thick, Thin | Any **one** |
| Standard Toppings | Tomato, Cheese | Standard, **compulsory** |
| Additional Toppings | Pepperoni, Chicken, Extra cheese  Mushrooms, Spinach, Olives | Upto **three** |

* Customers need to be able to **design, confirm/change/cancel** their pizza.
* Records are kept. They must store these properties:
  + Number of each base
  + Number of each size
  + Count of toppings sold
  + Type of toppings sold

## TASK 1

* The customer is given choices of **sizes**
  + The entry is validated [lookup check]
  + Unless the data is valid, an **error message** is printed and the customer is asked to re-enter
* The customer is given choices of **bases**
  + The entry is validated [lookup check]
  + Unless the data is valid, an error message is printed and the customer is asked to re-enter
* The customer is given choices of **number and type of toppings**
  + The entry is validated [lookup check, range check]
  + Unless the data is valid, an error message is printed and the customer is asked to re-enter
* The customer must be given three choices
  + Confirm their order
  + Alter their order
  + Not proceed
* If they confirm their order, they’re given a **unique order number**.

## TASK 2

Extend **TASK 1** to

* Record the number of
  + Each size of pizza sold
  + Each type of pizza base sold
  + Each type of toppings sold
* Calculate the **total number of pizzas ordered**

## TASK 3

Use the results of **TASK 2** to

* Display the **most popular toppings** as a **percentage** of the total toppings ordered
* Display the **least popular toppings** as a **percentage** of the total toppings ordered.

Write and test a program or programs for the pizza ordering service.

* Your program or programs must include appropriate prompts for the entry of data.
* Error messages and other output need to be set out clearly and understandably.
* All variables, constants and other identifiers must have meaningful names.

# Rough program structure – Version 1

* Setup **constant arrays** for the three options

SizesAvailable["Small", "Medium", "Large"]

BasesAvailable["Thick", "Thin"]

ToppingsAvailable["Pepperoni", "Chicken", "Extra Cheese", "Mushrooms", "Spinach", "Olives"]

* Setup **variable arrays** for counters

Sizes[1:3]; TotalSizes[1:3]

Bases[1:2]; TotalBases[1:2]

Toppings[1:6]; TotalToppings[1:6]

TASK 1

Input one validated order

* **Print** out the sets of **options**
* Ask customer to **input** **names** of items
  + **Validate** inputs
  + Increment a **counter of toppings**
  + Once **toppings exceeds 3** or customer says ‘Done’, move on.

**Update 1.1**

* Ask the customer to **input names** of items
* Input the number of toppings
* Input the names of items
* Validate the inputs
* Give customer **choices** to
  + Confirm the order
  + Alter the order
  + Not proceed
* If the order is **altered**, **input the new order**
* If the customer chooses to **not proceed**, **reset the system**
* If the order is **confirmed**, **generate a unique order ID**.
* **Output** the unique ID and a **summary** of the order
* **Loop** the system until no more pizzas are to be ordered.

## TASK 2

Record number of pizzas ordered

* Increment a counter of **number of pizzas** if an order is confirmed
* Add the value of the Counters[] to the TotalCounters[]
* **Output** the number of pizzas ordered and the numbers of each option ordered.

## TASK 3

Find the most and least popular toppings

* Calculate the highest ordered toppings
* Calculate the lowest ordered toppings
* Express both values as a percentage of the total orders

**Rough program structure – Version 2**

* Setup **constant arrays** for the three options

SizesAvailable["Small", "Medium", "Large"]

BasesAvailable["Thick", "Thin"]

ToppingsAvailable["Pepperoni", "Chicken", "Extra Cheese", "Mushrooms", "Spinach", "Olives"]

* Setup **variable arrays** for counters

Sizes[1:3]; TotalSizes[1:3]

Bases[1:2]; TotalBases[1:2]

Toppings[1:6]; TotalToppings[1:6]

* **Initialize all variable arrays** to have values 0

**Update 2.1**

* Segment D would be used instead of Segment A.

## TASK 1

Input one validated order

**Standard validation routine**

*Value* ← ""

WHILE (*Value* <> "*Choice*") ... AND ... (*Value* <> "*Choice*") DO

INPUT *Value*

IF (*Value* <> "*Choice*") ... AND ... (*Value* <> "*Choice*")

THEN PRINT *Error Message*

ENDIF

ENDWHILE

* **Print** out the sets of **options**
* Ask the customer to input the **size** of the pizza they would like

**Update 2.1**

* Flagging system is not to be used in validation
* *Counters*[] arrays are to be removed
* An array is to be used for an order OrderData[1:3+]
  + Validate the choice using a WHILE Loop
  + As long as the data is invalid
    - Output an **error message**
    - Ask the customer to re-enter the data from the list of sizes
  + If the data is valid, move on
* Ask the customer to input the **base** of the pizza they would like
  + Validate the choice using a WHILE Loop
  + As long as the data is invalid
    - Output an **error message**
    - Ask the customer to re-enter the data from the list of bases
  + If the data is valid, move on
* Ask the customer to input the number of toppings they would like
  + Validate the choice so that it is **a whole number where**
  + As long as the choice is invalid
    - Output an **error message**
    - Ask the customer to re-enter the data within the range
* Ask the customer to input **toppings** of the pizza they would like using a FOR Loop
  + Validate each choice using a WHILE Loop
  + As long as the data is invalid
    - Output an **error message**
    - Ask the customer to re-enter the data from a list of toppings
  + If the data is valid, move on

**Update 2.1**

* + Store each chosen topping in Toppings[]
* Give customer **choices** to
  + Confirm the order
  + Alter the order
  + Not proceed
* If the order is **altered**, **input the new order**
* If the customer chooses to **not proceed**, **reset the system**
* If the order is **confirmed**, **generate a unique order ID**.
* **Output** the unique ID
* **Input** from the staff whether any more pizzas are to be ordered
* **Loop** the system until no more pizzas are to be ordered.

TASK 2

Record number of pizzas ordered

* Increment a counter of **number of pizzas** if an order is confirmed
* Add the value of the Counters[] to TotalCounters[]
* **Output** the number of pizzas ordered and the numbers of each option ordered.

TASK 3

Find the most and least popular toppings

**Sample routine to find lowest**

Lowest ← 1000

FOR Count ← 1 TO *NumberOfChoices*

IF (*TotalCounter*[Count] < Lowest) AND (*TotalCounter*[Count] > 0)

THEN Lowest ← *TotalCounter*[Count]

ENDIF

NEXT Count

* Calculate the highest ordered toppings
* Calculate the lowest ordered toppings
* Express both values as a percentage of the total orders

# Test data

## Expected Results

|  |  |  |
| --- | --- | --- |
| Type of Data | Data | Expected result |
| Normal | Medium, Thin, 2, Spinach, **Olives**, Confirm, FALSE | 0 |
| Large, Thick, 2, **Olives**, **Chicken**, Confirm, FALSE | 1 |
| Small, Thin, 2, Spinach, Olives, Alter, Medium, Thin, 2, Spinach, **Olives**, Confirm, TRUE | 2 |
| [NO DATA] | 3  Olives, 50%  Chicken, 16.6% |
| Boundary | Large, Thin, 3, Chicken, Pepperoni, **Olives**, Confirm, FALSE | 0 |
| Large, Thick, 0, Confirm, FALSE | 1 |
| Small, Thin, 3, Extra Cheese, Mushrooms, Olives, Alter, Medium, Thin, 3, Mushrooms, Extra Cheese, **Olives**, Confirm, TRUE | 2 |
| [NO DATA] | 3  Olives, 33.3%  Pepperoni, 16.6% |
| Erroneous | Tiny, Fat, 6, Meat, Okay, NO | [ERRORS] |
| Huge, 0, Three, Chocolate, Fine, YES |

## Actual Results Version 1.0

|  |  |  |
| --- | --- | --- |
| Type of Data | Data | Actual result |
| Normal | Medium, Thin, 2, Spinach, **Olives**, Confirm, FALSE | **0** |
| Large, Thick, 2, **Olives**, **Chicken**, Confirm, FALSE | **1** |
| Small, Thin, 2, Spinach, Olives, Alter, Medium, Thin, 2, Spinach, **Olives**, Confirm, TRUE | **2** |
| [NO DATA] | **3**  **Olives**, **0%**  **Pepperoni**, **0%** |
| Boundary | Large, Thin, 3, Chicken, Pepperoni, **Olives**, Confirm, FALSE | **0** |
| Large, Thick, 0, Confirm, FALSE | **1** |
| Small, Thin, 3, Extra Cheese, Mushrooms, Olives, Alter, Medium, Thin, 3, Mushrooms, Extra Cheese, **Olives**, Confirm, TRUE | **2** |
| [NO DATA] | **3**  **Olives**, **0%**  **Pepperoni**, **0%** |
| Erroneous | Tiny, Fat, 6, Meat, Okay, NO | **[ERRORS]** |
| Huge, 0, Three, Chocolate, Fine, YES |

## Actual Results Version 1.1

|  |  |  |
| --- | --- | --- |
| Type of Data | Data | Actual result |
| Normal | Medium, Thin, 2, Spinach, **Olives**, Confirm, FALSE | **0** |
| Large, Thick, 2, **Olives**, **Chicken**, Confirm, FALSE | **1** |
| Small, Thin, 2, Spinach, Olives, Alter, Medium, Thin, 2, Spinach, **Olives**, Confirm, TRUE | **2** |
| [NO DATA] | **3**  **Olives**, **50%**  **Chicken**, **16.6%** |
| Boundary | Large, Thin, 3, Chicken, Pepperoni, **Olives**, Confirm, FALSE | **0** |
| Large, Thick, 0, Confirm, FALSE | **1** |
| Small, Thin, 3, Extra Cheese, Mushrooms, Olives, Alter, Medium, Thin, 3, Mushrooms, Extra Cheese, **Olives**, Confirm, TRUE | **2** |
| [NO DATA] | **3**  **Olives**, **33.3%**  **Pepperoni**, **16.6%** |
| Erroneous | Tiny, Fat, 6, Meat, Okay, NO | **[ERRORS]** |
| Huge, 0, Three, Chocolate, Fine, YES |

## Actual Results Version 2.1

|  |  |  |
| --- | --- | --- |
| Type of Data | Data | Actual result |
| Normal | Medium, Thin, 2, Spinach, **Olives**, Confirm, FALSE | **0** |
| Large, Thick, 2, **Olives**, **Chicken**, Confirm, FALSE | **1** |
| Small, Thin, 2, Spinach, Olives, Alter, Medium, Thin, 2, Spinach, **Olives**, Confirm, TRUE | **2** |
| [NO DATA] | **3**  **Olives**, **50%**  **Chicken**, **16.6%** |
| Boundary | Large, Thin, 3, Chicken, Pepperoni, **Olives**, Confirm, FALSE | **0** |
| Large, Thick, 0, Confirm, FALSE | **1** |
| Small, Thin, 3, Extra Cheese, Mushrooms, Olives, Alter, Medium, Thin, 3, Mushrooms, Extra Cheese, **Olives**, Confirm, TRUE | **2** |
| [NO DATA] | **3**  **Olives**, **33.3%**  **Pepperoni**, **16.6%** |
| Erroneous | Tiny, Fat, 6, Meat, Okay, NO | **[ERRORS]** |
| Huge, 0, Three, Chocolate, Fine, YES |

# Problems Encountered

## Problem 1

This problem was discovered after the results **Actual Results Version 1.0** did not match the expected results.

#### First observations

* **Normal test data**
  + The percentage of sales accounted by the most popular topping is reported **0%**
  + The least popular topping is reported **Pepperoni**
  + The percentage of sales accounted by the least popular topping is reported **0%**
* **Boundary test data**
  + The percentage of sales accounted by the most popular topping is reported **0%**
  + The least popular topping is reported **Pepperoni**
  + The percentage of sales accounted by the least popular topping is reported **0%**

#### Conclusions

The data required from **TASK 3** is wrong/incomplete.

* It appears that the program blindly classifies **Pepperoni** as the least popular topping.
* It also seems that the percentage of sales accounted by both, the least popular topping and the most popular topping, is either calculated **0%** or not calculated at all.

# Resolution of the Problems

## Problem 1

The problem was resolved when the results of **Actual Data Version 1.**1 matched the expected results.

The problem appears to be with the arrays involved calculations of **TASK 3**.

#### Diagnosis

* The TotalCount[] arrays were not initialized with values 0
* When checking for the least popular toppings, even the ones not sold at all [TotalCount[i] = 0] were being counted.

#### Fix

* The TotalCount[] arrays have been initialized with values 0 using FOR loops.
* When checking for the least popular toppings, the ones not sold at all are now ignored.

#### Changes

* Segment A was added
* Segment B was changed to Segment B

**Segment A**

// Initialize the array with all values 0

FOR Count ← 1 TO 3 // Iterate 3 times for 3 values

TotalSizes[Count] ← 0 // Write 0 to the current value

NEXT Count

// Initialize the array with all values 0

FOR Count ← 1 TO 2 // Iterate 2 times for 2 values

TotalBases[Count] ← 0 // Write 0 to the current value

NEXT Count

// Initialize the array with all values 0

FOR Count ← 1 TO 6 // Iterate 6 times for 6 values

TotalToppings[Count] ← 0 // Write 0 to the current value

NEXT Count

**Segment B**

// Calculate the lowest sales

IF TotalToppings[Count] < Lowest // If the current topping sold less than the running least popular topping

THEN

Lowest ← TotalToppings[Count] // Update the running least popular topping

LowestIndex ← Count // Record the array index of the topping

ENDIF

**Segment C**

IF (TotalToppings[Count] < Lowest) AND (TotalToppings[Count] > 0) // If the current topping sold less than the running least popular topping and it sold in the first place

THEN

Lowest ← TotalToppings[Count] // Update the running least popular topping

LowestIndex ← Count // Record the array index of the topping

ENDIF

**Segment D**

TotalSizes ← [0, 0, 0] // Set values for 3 sizes

TotalBases ← [0, 0] // Set values for 2 bases

TotalToppings ← [0, 0, 0, 0, 0, 0] // Set values for 6 toppings

# Identifiers

|  |  |  |
| --- | --- | --- |
| Pseudocode | Type | Purpose |
| SizesAvailable[1:3]  BasesAvailable[1:2]  ToppingsAvailable[1:6] | Constant String Array | Store the names of the options in the various attributes of the pizza |
| Sizes[1:3]  Bases[1:2]  Toppings[1:6] | Variable Boolean Array | Store whether a particular option has been ordered within an order |
| TotalSizes[1:3]  TotalBases[1:2]  TotalToppings[1:6] | Variable Integer Array | Count how many of each option has been ordered in total |
| OrderData[1:(2+)] | Dynamic Mutable Variable Array | Stores the items of the current order |
| Number of items | Variable Integer | Stores the number of items of the current order |
| MaxToppings | Constant Integer | The maximum number of toppings allowed |
| CurrentID | Variable Integer | Stores the running unique order ID |
| OrdersCount | Variable Integer | Stores the running count of confirmed orders |
| Close | Variable Boolean | Stores the status about ending the program |
| Highest  HighestIndex  Lowest  LowestIndex | Variable Real  Variable Integer | Store the highest and lowest sales of toppings |
| Count  CountI  CountO | Variable Integer | FOR loop index counters |
| Status | Variable String | Stores the current status of the order |
| Size  Base  Topping | Variable String | Store the data entered by the user |
| ToppingChoice | Variable Integer | Stores the number of toppings the user would like |

#### THOUGHTS

I have taken a long approach, perhaps unnecessarily detailed, to build this report along with my application. Nevertheless, I think I now understand the value of documentation. The detailed paraphrasing and listing of the design have allowed me to better understand my application. Using a detailed pseudocode before writing a Python program helped confirm the design of the application. The test data proved helpful in finding and fixing errors. I must also say that, rather surprisingly, I enjoyed building this report. Now as I complete this file, I am confident that the data would help me better prepare for my exams and achieve the A\* grade easily, perhaps with full marks, in IGCSE Computer Science 0478.

Anuj Verma

CLAD