```
START

// **   DECLARE CONSTANTS

// These are the options of the various properties of the pizza available
SizesAvailable["Small", "Medium", "Large"]  // The size of the pizza
BasesAvailable["Thick", "Thin"]  // The type of base of the pizza
ToppingsAvailable["Pepperoni", "Chicken", "Extra Cheese", "Mushrooms", "Spinach",
"Olives"]  // The toppings available

MaxToppings ← 3  // The maximum number of toppings that can be taken


// **   DECLARE VARIABLES
CurrentID ← 0  // The running unique ID of the order
OrdersCount ← 0  // The running total of the number of confirmed orders
Close ← FALSE  // Status of more orders

Highest ← 0
HighestIndex ← 0
Lowest ← 1000
LowestIndex ← 0
ToppingsSum ← 0

OrderData[]  // Running tracker of all the items of one order

TotalSizes[1:3]  // Running counter of the sizes taken
TotalBases[1:2]  // Running counter of the pizza bases taken
TotalToppings[1:6]  // Running counter of the toppings taken

// Initialize the array with all values 0
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

// **   TASK 1
// Use a default status "Alter" to customize the pizza
// Input the values of each attribute and validate them
// Give the customer a choice to alter the order, confirm it or cancel it
// If they choose to alter, re-input the values
// If they confirm it, provide them with a new order number.

// **   TASK 2
// 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.


REPEAT

    Status ← "Alter"  // Default status to input values

    // Input and validate the values
    WHILE Status = "Alter" DO  // As long as the status is "Alter"

        // Reset the running tracker
        OrderData[1:3]  // Initialize to have 0 toppings

        // Output the available options

        // Output the sizes
        PRINT "The following sizes are available to choose from:"
        FOR Count ← 1 TO 3  // Iterate 3 times for 3 sizes
            PRINT SizesAvailable[Count]  // Output the available sizes
        NEXT Count
        // Output the bases
        PRINT "The following bases are available to choose from:"
        FOR Count ← 1 TO 2  // Iterate 2 times for 2 pizza bases
            PRINT BasesAvailable[Count]  // Output the available bases
        NEXT Count

        // Output the toppings
        PRINT "The following toppings are available to choose from:"
        FOR Count ← 1 TO 6  // Iterate 6 times for 6 toppings
            PRINT ToppingsAvailable[Count]  // Output the available toppings
        NEXT Count

        //Input and validate the size of the pizza
        PRINT "Please enter the size of the pizza you would like:"  // Input prompt

        Size ← ""  // Enable the DO WHILE loop to run by making the size invalid

        WHILE (Size <> "Small") AND (Size <> "Medium") AND (Size <> "Large") DO  // Validation loop
            INPUT Size  // Input the (corrected) size

            IF (Size <> "Small") AND (Size <> "Medium") AND (Size <> "Large")  // If the size is invalid
                THEN PRINT "The size you have entered is invalid. Please re-enter the size from one of the options above:"  // Print error message and ask for correction
            ENDIF

        ENDWHILE  // Unless the size is invalid, break out of the loop

        //Input and validate the base of the pizza
        PRINT "Please enter the pizza base you would like:"  // Input prompt

        Base ← ""  // Enable the DO WHILE loop to run by making the base invalid

        WHILE (Base <> "Thick") AND (Base <> "Thin") DO  // Validation loop
            INPUT Base  // Input the corrected base

            IF (Base <> "Thick") AND (Base <> "Thin")  // If the base is invalid
                THEN PRINT "The base you have entered is invalid. Please re-enter the base from one of the options above:"  // Print error message and ask for correction
            ENDIF

        ENDWHILE  // Unless the base is invalid, break out of the loop

        // Input and validate the number of toppings the customer wants
        PRINT "How many toppings do you want on your pizza? You may enter any whole number between 0 and 3."  // Input prompt

        WHILE NOT ((ToppingChoice < 3) AND (ToppingChoice > 0)) DO  // Validation loop
            INPUT INTEGER ToppingChoice  // Input the number of toppings the user wants

            IF NOT ((ToppingChoice < 3) AND (ToppingChoice > 0)) // If the number of toppings is invalid
                THEN PRINT "You have entered an invalid number of toppings. Please re-enter any whole number between 0 and 3."  // Throw error message and ask for correction
            ENDIF

        ENDWHILE  // Unless the number of toppings is greater than 3, break out of the loop

        NumberOfItems ← 3 + ToppingChoice  // Calculate the total number of items based on the number of toppings
        OrderData[1:NumberOfItems]  // Declare an array with as many elements as in the order

```

```
            // Store the data acquired so far
            OrderData[1] ← Size  // Store the size
            OrderData[2] ← Base  // Store the base
            OrderData[3] ← NumberOfItems  // Store the total number of items

            FOR CountO ← 1 TO ToppingChoice  // Iterate as many times as the toppings
            taken

                //Input and validate the topping of the pizza
                PRINT "Please enter topping", (CountO + 1), "of the pizza you would
                like:"  // Input prompt

                Topping ← ""  // Enable the DO WHILE loop to run by making the topping
                invalid

                WHILE (Topping <> "Pepperoni") AND (Topping <> "Chicken") AND (Topping <>
                 "Extra Cheese") AND (Topping <> "Mushrooms") AND (Topping <> "Spinach")
                AND (Topping <> "Olives")  // Validation loop
                    INPUT Topping  // Input the corrected topping

                    IF (Topping <> "Pepperoni") AND (Topping <> "Chicken") AND (Topping
                    <> "Extra Cheese") AND (Topping <> "Mushrooms") AND (Topping <>
                    "Spinach") AND (Topping <> "Olives")  // If the topping is invalid
                        THEN PRINT "The topping you have entered is invalid. Please
                        re-enter the topping from one of the options above:"  // Print
                        error message and ask for correction
                    ENDIF

                ENDWHILE   // Unless the topping is invalid, break out of the loop

                OrderData[3 + CountO] ← Topping  // Store the validated topping in the
                array

            NEXT CountO  // Move on to the next topping

            // Allow the customer to choose whether they want to alter their order,
            confirm it or cancel it
            PRINT "Do you want to Alter your order, Confirm or Not proceed?"  // Input
            prompt
            INPUT Status  // Input whether the customer wants to alter their order,
            confirm it or cancel it

        UNTIL Status <> "Alter"  // Unless they want to alter their order, break out of
        the loop

        // Give the customer a unique order ID if they have confirmed it
        IF Status = "Confirm"  // If the customer has confirmed their order
        THEN
            PRINT "Your unique order number is:", CurrentID  // Print out the unique ID
            CurrentID ← CurrentID + 1  // Increment the ID for the next confirmed order
            OrdersCount ← OrdersCount + 1  // Increment the counter for confirmed orders

            // Record how many of each size has been ordered
            FOR Count ← 1 TO 3  // Iterate 3 times for 3 sizes
                IF OrderData[1] = SizesAvailable[Count]  // If a size is recorded
                    THEN TotalSizes[Count] ← TotalSizes[Count] + 1  // Increment the
                    counter
                ENDIF
            NEXT Count

            // Record how many of each pizza base has been ordered
            FOR Count ← 1 TO 2  // Iterate 2 times for 2 pizza bases
                IF OrderData[2] = BasesAvailable[Count]  // If a pizza base is recorded
                    THEN TotalBases[Count] ← TotalBases[Count] + 1  // Increment the
                    counter
                ENDIF
            NEXT Count
            // Record how many of each topping has been ordered
            FOR CountO ← 1 TO OrderData[3]  // Run as many times as the number of toppings taken
                FOR CountI ← 1 TO 6  // Iterate 6 times for 6 toppings
                    IF OrderData[CountO] = ToppingsAvailable[CountI]  // If a topping has been ordered
                        THEN TotalToppings[CountI] ← TotalToppings[CountI] + 1  // Increment the counter
                    ENDIF
                NEXT Count
            NEXT CountO

        ENDIF


        PRINT "Do you want to exit the program?"  // Input prompt
        INPUT BOOLEAN Close  // Ask the staff if all orders are done

    UNTIL Close = TRUE  // Break out of the loop unless more pizzas are to be ordered

    PRINT OrdersCount, "pizzas were ordered."  // Output how many pizzas were ordered

    // **   TASK 3
    // Calculate the total number of toppings ordered
    // Calculate the highest ordered toppings
    // Calculate the lowest ordered toppings
    // Express both values as a percentage of the total orders

    FOR Count ← 1 TO 6  // Iterate 6 times for 6 toppings
        ToppingsSum ← ToppingsSum + TotalToppings[Count]  // Add to the running total to calculate the sum

        // Calculate the highest sales
        IF TotalToppings[Count] > Highest  // If the current topping sold more than the running most popular topping
        THEN
            Highest ← TotalToppings[Count]  // Update the running most popular topping
            HighestIndex ← Count  // Record the array index of the topping
        ENDIF

        // Calculate the lowest sales
        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

    NEXT Count

    PRINT ToppingsAvailable[HighestIndex], "was the most popular topping and accounted for", ((Highest/ToppingsSum) * 100), "% of the toppings sales."  // Output the most popular toppings
    PRINT ToppingsAvailable[LowestIndex], "was the least popular topping and accounted for", ((Lowest/ToppingsSum) * 100), "% of the toppings sales."  // Output the least popular toppings

    // This is the end of the program
    // All required tasks have been completed.

    END

```