```
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

Sizes[1:3]  // Running tracker of the size taken in an order
Bases[1:2]  // Running tracker of the pizza base taken in an order
Toppings[1:6]  // Running tracker of the toppings taken in an 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
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

// **   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 trackers
        Sizes ← [FALSE, FALSE, FALSE]
        Bases ← [FALSE, FALSE]
        Toppings ← [FALSE, FALSE, FALSE, FALSE, FALSE, FALSE]
```

```
71
72          // Output the available options
73
74          // Output the sizes
75          PRINT "The following sizes are available to choose from:"
76          FOR Count ← 1 TO 3  // Iterate 3 times for 3 sizes
77              PRINT SizesAvailable[Count]  // Output the available sizes
78          NEXT Count
79
80          // Output the bases
81          PRINT "The following bases are available to choose from:"
82          FOR Count ← 1 TO 2  // Iterate 2 times for 2 pizza bases
83              PRINT BasesAvailable[Count]  // Output the available bases
84          NEXT Count
85
86          // Output the toppings
87          PRINT "The following toppings are available to choose from:"
88          FOR Count ← 1 TO 6  // Iterate 6 times for 6 toppings
89              PRINT ToppingsAvailable[Count]  // Output the available toppings
90          NEXT Count
91
92          //Input and validate the size of the pizza
93          PRINT "Please enter the size of the pizza you would like:"  // Input prompt
94          INPUT Size  // Input the size
95
96          SizeValid ← FALSE  // Set flag as invalid
97
98          // Check if the size is valid
99          FOR Count ← 1 TO 3  // Iterate 3 times for 3 sizes
100             IF Size = SizesAvailable[Count]  // If a match is found from the
                available sizes
101             THEN
102                 SizeValid ← TRUE  // Set flag as valid
103                 Sizes[Count] ← TRUE  // Set flag as selected
104             ENDIF
105         NEXT Count
106
107         WHILE SizeValid = FALSE DO  // Validation loop
108             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
109             INPUT Size  // Input the corrected size
110
111             // Check if the size is valid
112             FOR Count ← 1 TO 3  // Iterate 3 times for 3 sizes
113                 IF Size = SizesAvailable[Count]  // If a match is found from the
                    available sizes
114                 THEN
115                     SizeValid ← TRUE  // Set flag as valid
116                     Sizes[Count] ← TRUE  // Set flag as selected
117                 ENDIF
118             NEXT Count
119
120         ENDWHILE   // Unless the size is invalid, break out of the loop
121
122         //Input and validate the base of the pizza
123         PRINT "Please enter the pizza base you would like:"  // Input prompt
124         INPUT Base  // Input the base
125
126         BaseValid ← FALSE  // Set flag as invalid
127
128         // Check if the base is valid
129         FOR Count ← 1 TO 2  // Iterate 2 times for 2 bases
130             IF BaseValid = BasesAvailable[Count]  // If a match is found from the
                available bases
131             THEN
132                 BaseValid ← TRUE  // Set flag as valid
133                 Bases[Count] ← TRUE  // Set flag as selected
134             ENDIF
135         NEXT Count
136
```

```
137         WHILE BaseValid = FALSE DO  // Validation loop
138             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
139             INPUT Base  // Input the corrected base
140
141             // Check if the base is valid
142             FOR Count ← 1 TO 2  // Iterate 2 times for 2 sizes
143                 IF Base = BasesAvailable[Count]  // If a match is found from the
                    available bases
144                 THEN
145                     BaseValid ← TRUE  // Set flag as valid
146                     Bases[Count] ← TRUE  // Set flag as selected
147                 ENDIF
148             NEXT Count
149
150         ENDWHILE   // Unless the base is invalid, break out of the loop
151
152         // Input and validate the number of toppings the customer wants
153         PRINT "How many toppings do you want on your pizza? You may enter any whole
            number 0 and 3."  // Input prompt
154         INPUT ToppingChoice  // Input the number of toppings the customer wants
155
156         ToppingChoiceValid ← FALSE  // Set flag as invalid
157
158         IF (ToppingChoice < 3) AND (ToppingChoice > 0)  // If the number of toppings
            is in the acceptable range
159             THEN ToppingChoiceValid ← TRUE  // Set flag as valid
160         ENDIF
161
162         WHILE ToppingChoiceValid = FALSE DO  // Validation loop
163             PRINT "You have entered an invalid number of toppings. Please re-enter
                any whole number 0 and 3."  // Throw error message and ask for correction
164             INPUT ToppingChoice
165
166             IF (ToppingChoice < 3) AND (ToppingChoice > 0)  // If the number of
                toppings is in the acceptable range
167                 THEN ToppingChoiceValid ← TRUE  // Set flag as valid
168             ENDIF
169
170         ENDWHILE  // Unless the number of toppings is greater than 3, break out of
            the loop
171
172         FOR CountO ← 1 TO ToppingChoice  // Iterate as many times as the toppings
            taken
173
174             //Input and validate the topping of the pizza
175             PRINT "Please enter topping", (CountO + 1), "of the pizza you would
                like:"  // Input prompt
176             INPUT Topping  // Input the topping
177
178             ToppingValid ← FALSE  // Set flag as invalid
179
180             // Check if the topping is valid
181             FOR CountI ← 1 TO 6  // Iterate 6 times for 6 toppings
182                 IF Topping = ToppingsAvailable[CountI]  // If a match is found from
                    the available toppings
183                 THEN
184                     ToppingValid ← TRUE  // Set flag as valid
185                     Toppings[CountI] ← TRUE  // Set flag as selected
186                 ENDIF
187             NEXT CountI
188
189             WHILE ToppingValid = FALSE  // Validation loop
190                 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
191                 INPUT Topping  // Input the corrected topping
192
193
194
```

```
                    // Check if the topping is valid
                    FOR Count ← 1 TO 6  // Iterate 6 times for 6 toppings
                        IF Topping = ToppingsAvailable[CountI]  // If a match is found from the available toppings
                        THEN
                            ToppingValid ← TRUE  // Set flag as valid
                            Toppings[CountI] ← TRUE  // Set flag as selected
                        ENDIF
                    NEXT CountI

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

            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 Sizes[Count] = TRUE  // 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 Bases[Count] = TRUE  // 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 Count ← 1 TO 6  // Iterate 6 times for 6 toppings
                IF Toppings[Count] = TRUE  // If a topping has been ordered
                    THEN TotalToppings[Count] ← TotalToppings[Count] + 1  // Increment the counter
                ENDIF
            NEXT Count

        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

```

```
258   FOR Count ← 1 TO 6  // Iterate 6 times for 6 toppings
259       ToppingsSum ← ToppingsSum + TotalToppings[Count]  // Add to the running total to
          calculate the sum
260
261       // Calculate the highest sales
262       IF TotalToppings[Count] > Highest  // If the current topping sold more than the
          running most popular topping
263       THEN
264           Highest ← TotalToppings[Count]  // Update the running most popular topping
265           HighestIndex ← Count  // Record the array index of the topping
266       ENDIF
267
268       // Calculate the lowest sales
269       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
270       THEN
271           Lowest ← TotalToppings[Count]  // Update the running least popular topping
272           LowestIndex ← Count  // Record the array index of the topping
273       ENDIF
274
275   NEXT Count
276
277   PRINT ToppingsAvailable[HighestIndex], "was the most popular topping and accounted
      for", ((Highest/ToppingsSum) * 100), "% of the toppings sales."  // Output the most
      popular toppings
278   PRINT ToppingsAvailable[LowestIndex], "was the least popular topping and accounted
      for", ((Lowest/ToppingsSum) * 100), "% of the toppings sales."  // Output the least
      popular toppings
279
280   // This is the end of the program
281   // All required tasks have been completed.
282
283   END
284
```