# Computer Science
## 2210/22

This document covers every aspect of Pre-Release Material including detailed explanations, Pseudocodes, efficiencies and expected questions.

## PRE-RELEASE MATERIAL
## OCT/NOV
## 2020

0335-1400368 @gilanihaseeb Syed Haseeb Bari Gilani CS/IT - O/A Level

# Pre-Release Material:

Your preparation for the examination should include attempting the following practical tasks by **writing and testing a program or programs**.

An online computer shop sells customised personal computers. Every computer sold includes a basic set of components costing $200 and additional items can be added from the table:

| Category | Item code | Description | Price ($) |
|---|---|---|---|
| Case | A1 | Compact | 75.00 |
| Case | A2 | Tower | 150.00 |
| RAM | B1 | 8 GB | 79.99 |
| RAM | B2 | 16 GB | 149.99 |
| RAM | B3 | 32 GB | 299.99 |
| Main Hard Disk Drive | C1 | 1 TB HDD | 49.99 |
| Main Hard Disk Drive | C2 | 2 TB HDD | 89.99 |
| Main Hard Disk Drive | C3 | 4 TB HDD | 129.99 |
| Solid State Drive | D1 | 240 GB SSD | 59.99 |
| Solid State Drive | D2 | 480 GB SSD | 119.99 |
| Second Hard Disk Drive | E1 | 1 TB HDD | 49.99 |
| Second Hard Disk Drive | E2 | 2 TB HDD | 89.99 |
| Second Hard Disk Drive | E3 | 4 TB HDD | 129.99 |
| Optical Drive | F1 | DVD/Blu-Ray Player | 50.00 |
| Optical Drive | F2 | DVD/Blu-Ray Re-writer | 100.00 |
| Operating System | G1 | Standard Version | 100.00 |
| Operating System | G2 | Professional Version | 175.00 |

As well as the basic set of components every computer must include one case, one RAM and one Main Hard Disk Drive from the table.
A computer is supplied with or without an Operating System.

Write and test a program or programs for the online computer shop.

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

You will need to complete these **three** tasks. Each task must be fully tested.

**Task 1** – Setting up the system and ordering the main items.
Write a program to:

- use arrays to store the item code, description and price
- allow a customer to choose one case, one RAM and one Main Hard Disk Drive
- calculate the price of the computer using the cost of the chosen items and the basic set of components
- store and output the chosen items and the price of the computer.

**Task 2** – Ordering additional items.
Extend TASK 1 to:

- allow a customer to choose whether to purchase any items from the other categories – if so, which item(s)
- update the price of the computer
- store and output the additional items and the new price of the computer.

**Task 3** – Offering discounts.
Extend TASK 2 to:

- apply a 5% discount to the price of the computer if the customer has bought only one additional item
- apply a 10% discount to the price of the computer if the customer has bought two or more additional items
- output the amount of money saved and the new price of the computer after the discount.

## Main Idea of Pre-Release Material:

- This pre-release material contains a table with information of different components for a computer, 3 tasks and a few instructions.
- The table contains categories, item codes, descriptions and prices of different components that can be bought by the customer.
- It is based on an online computer shop which sells customised personal computers. Every computer contains basic components costing $200.
- A customer must buy one case, one RAM and one Main Hard Disk Drive according to his/her choice from the table. The price will be calculated, details of items will be stored and both will be displayed.
- A customer can also buy additional items from the table. The cost for additional items will be added to the total price, details of additional items stored and information will be updated. Both the details of additional items and prices will be displayed.
- Furthermore, a customer will be given 5% to 10% discount depending upon the number of additional items bought. The money saved by the customer, newly discounted total price will be stored and both will be displayed.

## Explanation of Pre-Release Material:

Your preparation for the examination should include attempting the following practical tasks by **writing and testing a program or programs**.

An online computer shop sells customised personal computers. Every computer sold includes a basic set of components costing $200 and additional items can be added from the table:

**It can be understood from this piece of text that:**

- initial price of every computer is $200.
- additional items can be added in the computer according to customers choice.

17 total items

A unique code and description for identification of every single item

| | Category | Item code | Description | Price ($) |
|---|---|---|---|---|
| 2 cases | Case | A1 | Compact | 75.00 |
| | Case | A2 | Tower | 150.00 |
| 3 RAMs | RAM | B1 | 8 GB | 79.99 |
| | RAM | B2 | 16 GB | 149.99 |
| | RAM | B3 | 32 GB | 299.99 |
| 3 Main HDD | Main Hard Disk Drive | C1 | 1 TB HDD | 49.99 |
| | Main Hard Disk Drive | C2 | 2 TB HDD | 89.99 |
| | Main Hard Disk Drive | C3 | 4 TB HDD | 129.99 |
| 2 SSD | Solid State Drive | D1 | 240 GB SSD | 59.99 |
| | Solid State Drive | D2 | 480 GB SSD | 119.99 |
| 3 Second HDD | Second Hard Disk Drive | E1 | 1 TB HDD | 49.99 |
| | Second Hard Disk Drive | E2 | 2 TB HDD | 89.99 |
| | Second Hard Disk Drive | E3 | 4 TB HDD | 129.99 |
| 2 Optical | Optical Drive | F1 | DVD/Blu-Ray Player | 50.00 |
| | Optical Drive | F2 | DVD/Blu-Ray Re-writer | 100.00 |
| 2 OS | Operating System | G1 | Standard Version | 100.00 |
| | Operating System | G2 | Professional Version | 175.00 |

As well as the basic set of components every computer must include one case, one RAM and one Main Hard Disk Drive from the table.
A computer is supplied with or without an Operating System.

**It can be understood from this piece of text that:**

- every computer is to compulsorily include three items (in any case):
    1. one case (out of 2 given in table)
    2. one RAM (out of 3 given in table)
    3. one main HDD (out of 3 given in table)
- an operating system may or may not be bought by the customer (it's not compulsory)

Write and test a program or programs for the online computer shop.
- Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry.
- Error messages and other output need to be set out clearly and understandably.
- All arrays, variables, constants and other identifiers must have meaningful names.

**It can be understood from this piece of text that:**

- the code must contain formal, suitable and clearly understandable messages/prompts that must be displayed when asking for input of data.
- the data must be validated through various checks and using selection statements ***(IF..THEN..END IF)*** and conditional loops ***(WHILE..DO..END WHILE)***
- if the input is wrong then the error message must be displayed and it should be formal, suitable and clearly understandable as well.
- all output of data must be displayed with proper messages/prompts describing what is the output showing or telling. They should be formal, suitable and clearly understandable as well.
- the program will use a number of arrays, variables and constants which must have clearly understandable and meaningful names that makes sense. (instead of using names such as $price, the meaningful name must be used such as ***total_price*** etc.

**By now you must be clear about following points:**

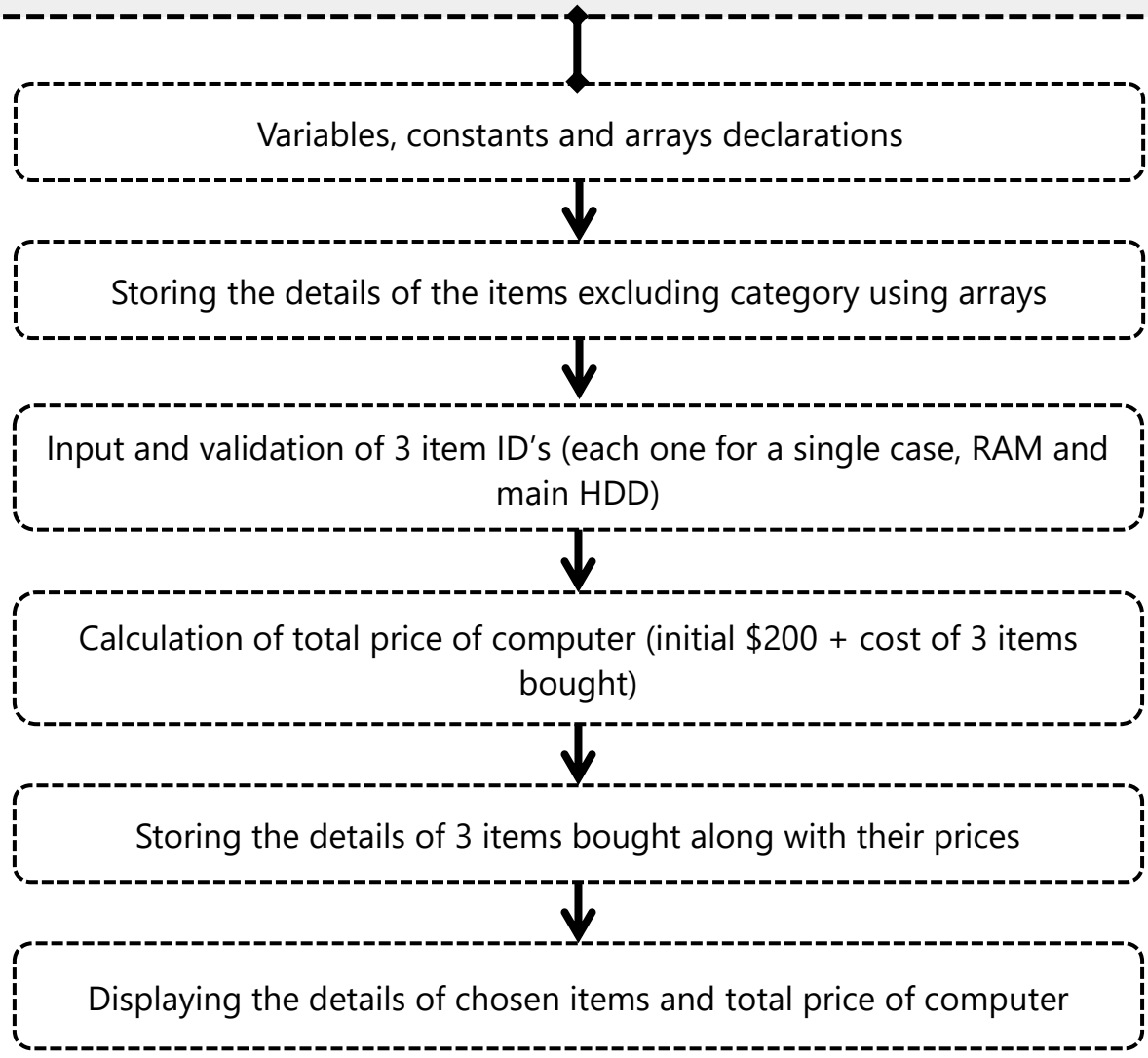
- A customer can buy a customized personal computer.
- It contains basic set of components worth $200.
- The customer must buy one case, one RAM and one main HDD (3 out of 8 items in total).
- The customer can buy additional items which have different prices as well.
- The code must display proper messages for input, error or output.
- The code must validate every input data.
- The code must use arrays, variables and constants (wherever required) but they should be properly named so they make sense to the examiner.

## Concept and understanding of TASK 1:

**Task 1** – Setting up the system and ordering the main items.

Write a program to:

- use arrays to store the item code, description and price
- allow a customer to choose one case, one RAM and one Main Hard Disk Drive
- calculate the price of the computer using the cost of the chosen items and the basic set of components.
- store and output the chosen items and the price of the computer.



## Explanation of Algorithm of TASK 1:

In this task, we have to calculate the total price of a computer after purchasing of a few components/items.

- use arrays to store the item code, description and price

We will make use of **1D arrays to store the information** given in the table (item code, description and prices) for all 17 items like this:

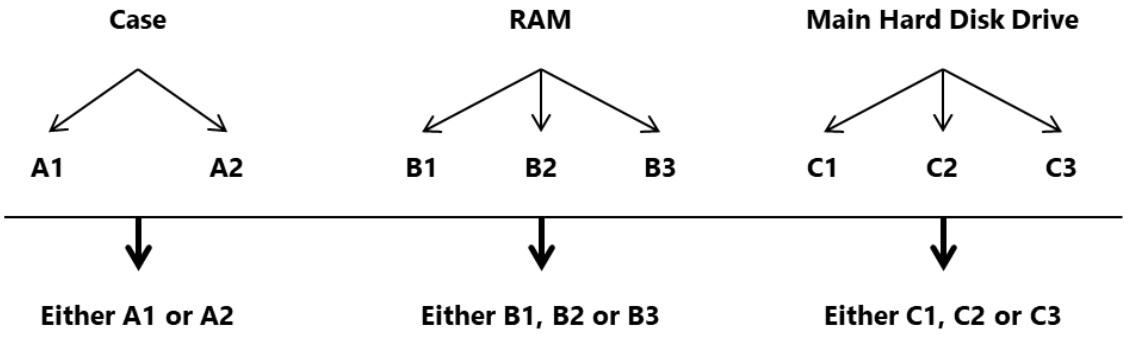
| *Index* | item_code | item_description | item_price |
|---|---|---|---|
| **1** | A1 | Compact | 75.00 |
| **2** | A2 | Tower | 150.00 |
| **3** | B1 | 8 GB | 79.00 |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| **17** | G2 | Professional Version | 175.00 |

An array for the column "Category" is not made since it is not required.

- allow a customer to choose one case, one RAM and one Main Hard Disk Drive

**Since only one case, one ram and one main HDD has to be chosen for this task, we will, for now, only consider this portion of the table which contains 8 items.**

| | Category | Item code | Description | Price ($) |
|---|---|---|---|---|
| 1. | Case | A1 | Compact | 75.00 |
| 2. | Case | A2 | Tower | 150.00 |
| 3. | RAM | B1 | 8 GB | 79.99 |
| 4. | RAM | B2 | 16 GB | 149.99 |
| 5. | RAM | B3 | 32 GB | 299.99 |
| 6. | Main Hard Disk Drive | C1 | 1 TB HDD | 49.99 |
| 7. | Main Hard Disk Drive | C2 | 2 TB HDD | 89.99 |
| 8. | Main Hard Disk Drive | C3 | 4 TB HDD | 129.99 |



For the customer to choose these items, he/she would firstly need to have a look at those items along with their details. In order to achieve this, we will display/output details of those items category wise using a ***FOR*** loop.

First we will output the details of both cases using a ***FOR*** loop like this:

```
FOR count ← 1 TO 2
    PRINT "Item code: ", item_code[count]
    PRINT "Item description: ", item_description[count]
    PRINT "Item price: ", item_price[count]
NEXT count
```

**Running of example code:**

**When count will be 1:**

```
PRINT "Item code: ", item_code[1]
PRINT "Item description: ", item_description[1]
PRINT "Item price: ", item_price[1]
```

**Output will be:**

```
Item code: A1
Item description: Case Compact
Item price: 75.00
```

*Pre-defined values of item code, description and price stored in arrays (at the beginning of the code):*

item_code[1] ← "A1"
item_description[1] ← "Case Compact"
item_price[1] ← 75.00

**When count will be 2:**

```
PRINT "Item code: ", item_code[2]
PRINT "Item description: ", item_description[2]
PRINT "Item price: ", item_price[2]
```

**Output will be:**

```
Item code: A2
Item description: Case Tower
Item price: 150.00
```

*Pre-defined values of item code, description and price stored in arrays (at the beginning of the code):*

item_code[2] ← "A2"
item_description[2] ← "Case Tower"
item_price[2] ← 150.00

The customer will then choose the desired case by entering the item code of that case. A ***WHILE*** loop will be used for validation and to ensure that only item codes "A1" or "A2" are being entered by the customer. Input of any other item codes at this specific stage will output an error message.

In the similar way, the details of all three RAMs will be displayed using a ***FOR*** loop but with the following update:

```
FOR count ← 3 TO 5
```

**Running of example code:**

**When count will be 3:**

```
PRINT "Item code: ", item_code[3]
PRINT "Item description: ", item_description[3]
PRINT "Item price: ", item_price[3]
```

**Output will be:**

```
Item code: B1
Item description: RAM 8 GB
Item price: 79.99
```

*Pre-defined values of item code, description and price stored in arrays (at the beginning of the code):*

```
item_code[3] ← "B1"
item_description[3] ← "RAM 8 GB"
item_price[3] ← 79.99
```

A ***WHILE*** loop will be used for validation and to ensure that only item codes "B1" or "B2" or "B3" are being entered by the customer. Input of any other item codes at this specific stage will output an error message.

In the similar way, the details of all three Main Hard Disk Drives will be displayed using a ***FOR*** loop but with the following update as done previously:

```
FOR count ← 6 TO 8
```

The rest of the procedure is same as the one used for selection of cases and RAMs.

- calculate the price of the computer using the cost of the chosen items and the basic set of components.
- store and output the chosen items and the price of the computer.

First we have to store the details of chosen items in separate variables and then accordingly calculate the price of the computer using the cost of those items. Then we will add the cost of those items and the initial $200 for basic set of components to work out the total price.

There are two possible ways to store the details of chosen items with their prices. We will discuss and explain both the methods. The more efficient and easier one will be implemented.

The variables used for input of case, RAM, and main HDD item codes were as following:

***1) case_code 2) ram_code 3) main_disk_code***

**1) One method is to individually store details of every chosen item code in their specific variables using *IF…THEN…END IF* selection statements:**

First we will input the ***case_code***, validate it using ***WHILE*** loop and use ***IF*** statement for storing case details.

```
IF case_code = "A1" THEN
    case_price ← item_price[1], case_description ← item_description[1]
END IF
IF case_code = "A2" THEN
    case_price ← item_price[2], case_description ← item_description[2]
END IF
```

Then we will input the ***ram_code***, validate it using ***WHILE*** loop and use ***IF*** statement for storing RAM details. Similar statements will be written for B2 and B3 as well.

```
IF ram_code = "B1" THEN
    case_price ← item_price[3], case_description ← item_description[3]
END IF
```

At last we will input the ***main_disk_code***, validate it using ***WHILE*** loop and use ***IF*** statement for storing Main Hard Disk Drive details. Similar statements will be written for C1 and C2 as well.

```
IF main_disk_code = "C3" THEN
    case_price ← item_price[8], case_description ← item_description[8]
END IF
```

This method may seem easier to understand but it will consume a lot of time and space.

**2) Second method is to store details of every chosen item code in their specific variables using *FOR…TO…NEXT* loop:**

We would have taken the inputs in the code before this loop. Unlike the previous method, this code will not be used while taking inputs of every variable.

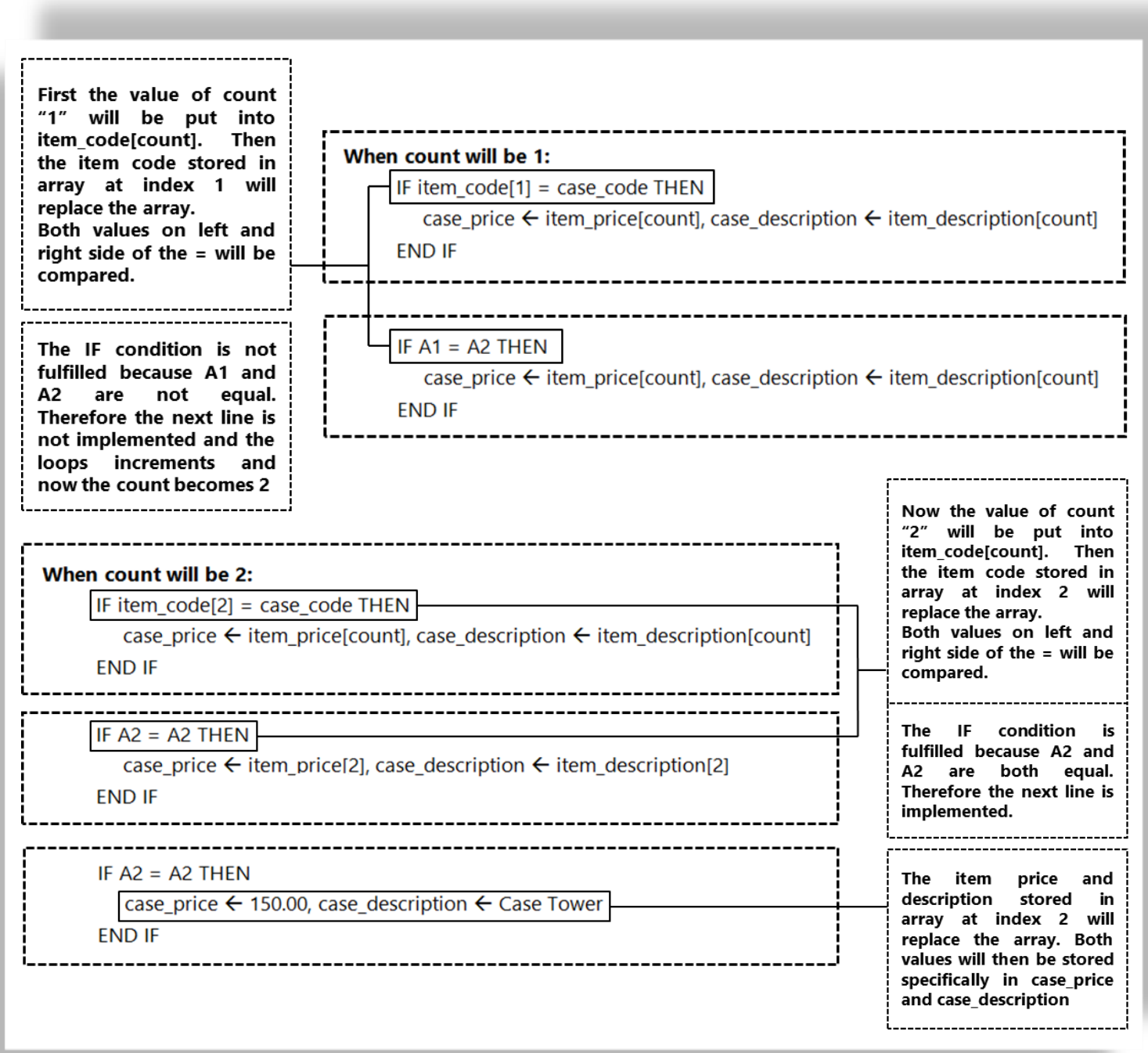
We will give this loop a count of 1 to 8 so that it runs for all the 8 items which are possible choices for the customer.

```
FOR count ← 1 TO 8
        IF item_code[count]  = case_code THEN
            case_price ← item_price[count],  case_description ← item_description[count]
        END IF
```

Now this may seem confusing at first but after understanding the running example of this code and the process that is being carried out, it will seem the most efficient and time saving method.

**Running of example code:**

**Suppose the customer wanted to choose the second case called "A2" instead of the first case called "A1". So he gave the input "A2" for the *case_code*.**




Once the details of case are stored, the loop will increment again. The counts from 1 to 2 were for **cases**. Understanding and keenly observing the process above will clear your mind on how the details are stored for only one case out of the two cases.

The counts from 3 to 5 are for **RAM**. In the similar manner, when the count will be 3 it will give the value B1. If it matches to the ***ram_code*** input given by user then all details stored in array at index 3 will be stored in the ***ram_description*** and ***ram_price***. If both values do not match then the loop will increment again. Ultimately out of B1, B2 and B3, any one value will match and its details will be stored.

The counts from 6 to 8 are for **Main Hard Disk Drives**. In the similar manner, when the count will be 6 it will give the value C1. If it matches to the ***main_disk_code*** input given by user then all details stored in array at index 6 will be stored in the ***main_disk_description*** and ***main_disk_price***. If both values do not match then the loop will increment again. Ultimately out of C1, C2 and C3, any one value will match and its details will be stored.

**The second method is more efficient and time saving so we will be using that approach, piece of code and logic in our Pseudocode for TASK 1.**

- calculate the price of the computer using the cost of the chosen items and the basic set of components.

Now we have stored the details of chosen items along with their prices. The final step is to calculate the **total price** of the computer and then output details with **total price**.

Calculation of **total price** is very easy and precise. We will simply add up the **initial price** of $200 with the prices of **case**, **RAM** and **Main Hard Disk Drive** which are stored in variables ***case_price, ram_price and main_disk_price***.

**(Remember that after addition of the prices of items, it is very important to add the total of items with the initial cost of basic set of components)**

- store and output the chosen items and the price of the computer.

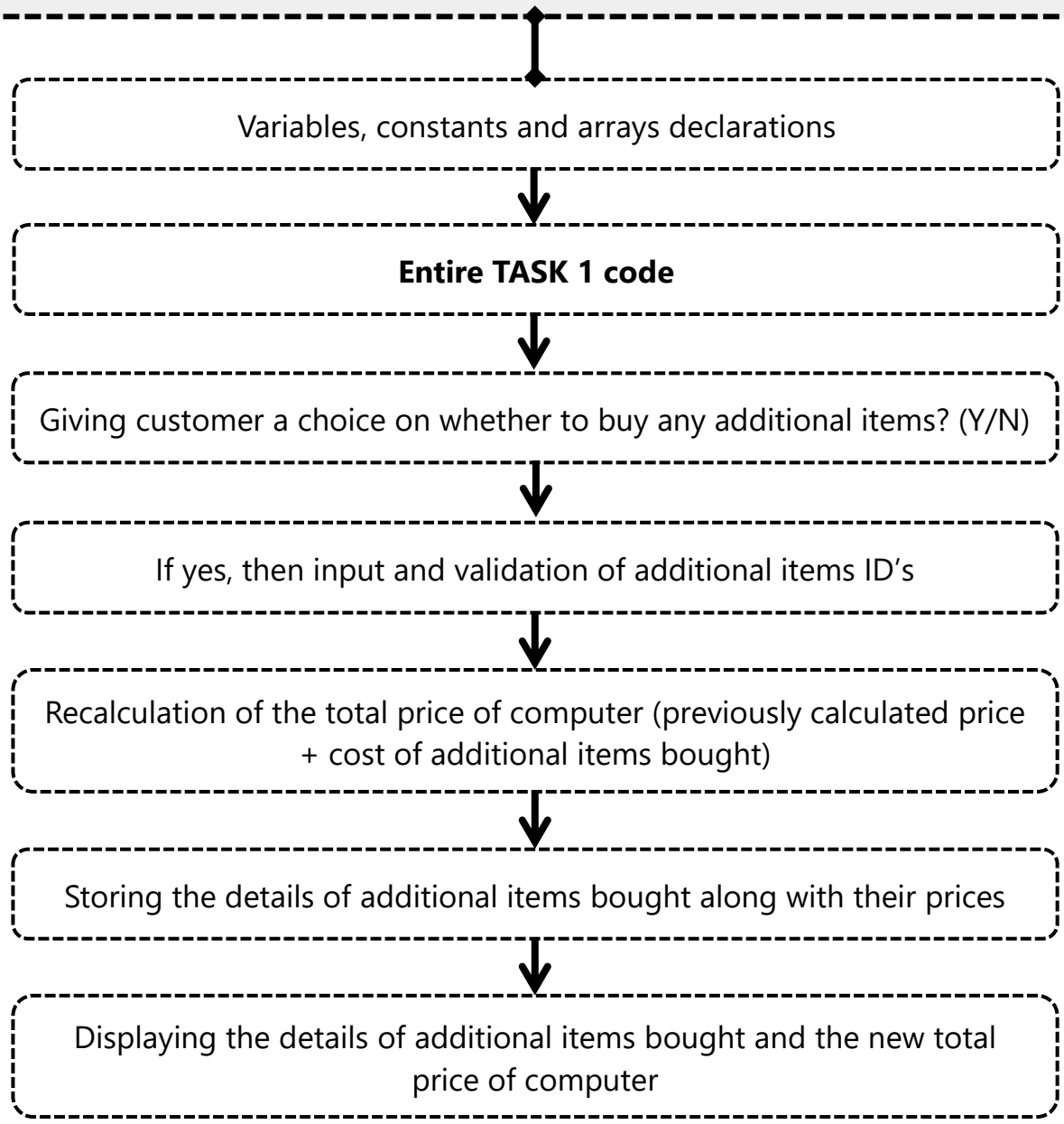
In the end, we will finally **PRINT the required output** only (as we have already stored the details of items and price of computer).

## Concept and understanding of TASK 2:

**Task 2** – Ordering additional items.

Extend TASK 1 to:

- allow a customer to choose whether to purchase any items from the other categories – if so, which item(s)
- update the price of the computer
- store and output the additional items and the new price of the computer.



## Explanation of Algorithm of TASK 2:

In this task, we have to give customer an option if he/she wants to buy any additional items and then recalculate the total price of a computer after purchasing of those additional items.

Extend TASK 1 to:

This means that the **identifiers and Pseudocode used in TASK 1** will be used for TASK 2 just like they are but with few extensions. TASK 2 will simply **extend** the conditions/demands of **TASK 1** and then finish.

- allow a customer to choose whether to purchase any items from the other categories – if so, which item(s)

**Since any additional item can be chosen for this task (other than the first 8 items), we will, for now, only consider this portion of the table which contains the last 9 items.**

| | Category | Item code | Description | Price ($) |
|---|---|---|---|---|
| 9. | Solid State Drive | D1 | 240GB SSD | 59.99 |
| 10. | Solid State Drive | D2 | 480GB SSD | 119.99 |
| 11. | Second Hard Disk Drive | E1 | 1TB HDD | 49.99 |
| 12. | Second Hard Disk Drive | E2 | 2TB HDD | 89.99 |
| 13. | Second Hard Disk Drive | E3 | 4TB HDD | 129.99 |
| 14. | Optical Drive | F1 | DVD/Blu-Ray Player | 50.00 |
| 15. | Optical Drive | F2 | DVD/Blu-Ray Re-writer | 100.00 |
| 16. | Operating System | G1 | Standard Version | 100.00 |
| 17. | Operating System | G2 | Professional Version | 175.00 |

Firstly, an appropriate output message will be displayed asking the customer if he/she wants to buy any additional items. A choice (Y/N) will be given to the customer and this input of choice will be validated on entry using ***WHILE*** loop. Any other input other than Y or N will output an error message.

If the customer wishes to buy additional items then he/she would firstly need to have a look at those items along with their details. In order to achieve this, we will display/output details of those items category wise using a ***FOR*** loop.
We will output the details of all the items at once using a ***FOR*** loop like this:

```
FOR count ← 9 TO 17
    PRINT "Item code: ", item_code[count]
    PRINT "Item description: ", item_description[count]
    PRINT "Item price: ", item_price[count]
NEXT count
```

**(Remember that the count will start from 9 and not 1. This is because the first 1-8 items have already been chosen by the customer in TASK 1. Now only the remaining items from 9 to 17 are available)**

**Running of example code:**

**When count will be 9:**

```
PRINT "Item code: ", item_code[9]
PRINT "Item description: ", item_description[9]
PRINT "Item price: ", item_price[9]
```

**Output will be:**

```
Item code: D1
Item description: Solid State Drive 240 GB SSD
Item price: 59.99
```

*Pre-defined values of item code, description and price stored in arrays (at the beginning of the code):*

item_code[9] ← "D1"
item_description[9] ← "Solid State Drive 240 GB SSD"
item_price[9] ← 59.99

**When count will be 13:**

```
PRINT "Item code: ", item_code[13]
PRINT "Item description: ", item_description[13]
PRINT "Item price: ", item_price[13]
```

**Output will be:**

```
Item code: E3
Item description: Hard Disk Drive 4 TB HDD
Item price: 129.99
```

*Pre-defined values of item code, description and price stored in arrays (at the beginning of the code):*

item_code[13] ← "E3"
item_description[13] ← "Hard Disk Drive 4 TB HDD"
item_price[13] ← 129.99

**When count will be 17:**

```
PRINT "Item code: ", item_code[17]
PRINT "Item description: ", item_description[17]
PRINT "Item price: ", item_price[17]
```

**Output will be:**

```
Item code: G2
Item description: Operating System Professional Version
Item price: 175.00
```

*Pre-defined values of item code, description and price stored in arrays (at the beginning of the code):*

item_code[17] ← "G2"
item_description[17] ← "Operating System Professional Version"
item_price[17] ← 175.00

The same method of giving outputs for additional items is used for every other item not covered above in running of example code such as "D2", "E1", "E2", "F1", "F2", and "G1".

After this, the customer would be asked about how many additional items they want to buy. This input of number of items will be validated using ***WHILE*** loop to ensure that this value is greater than 1.

The variable used for input of number of additional items customer wants to buy is: ***n***
The array used for input of additional items codes is: ***new_item_code[1:n]***

***(so that the array for new item code has as many indexes as the number of items to be bought (n))***

Since for the **TASK 1**, count was used in ***FOR*** loop for iterations. In **TASK 2**, we will use a separate variable called ***new_count*** in ***FOR*** loop for iterations.

The customer will then choose the desired item(s) by entering the item code(s). A ***WHILE*** loop will be used for validation and to ensure that only remaining item codes "D1", "D2", "E1", "E2", "E3", "F1", "F2", "G1" or "G2" are being entered by the customer. Input of any other item codes at this specific stage will output an error message.

All the additional item codes entered by the customer will be stored in ***new_item_code[1:n]*** array. The indexes of the array will be ***n*** so that it stores all the number of items to be bought. ***FOR*** loop will be used for entry of data as well since it is being stored in an array like this:

**Suppose the customer wants to buy 6 additional items. He will give the input of additional items (n) as 6.**

**The FOR loop will run from 1 to n (which is the number of additional items to be bought). Therefore in this case, it will run from 1 to 6.**

It is 6 in this case. So this piece of code will be iterated 6 times.

```
FOR new_count ← 1 to n
    PRINT "Enter the additional item(s) code."
        INPUT new_item_code[new_count]
```

```
FOR new_count ← 1 to 6
    PRINT "Enter the additional item(s) code."
        INPUT new_item_code[1]
```

**The array of new_item_code will be updated as the loops keeps updating**

| *new_count* | new_item_code |
|---|---|
| 1 | .... |
| 2 | .... |
| 3 | .... |
| 4 | .... |
| 5 | .... |
| 6 | .... |

**Running of example code:**

**Suppose the customer gives the following inputs of new item codes: D2, E1, E2, E3, F2 and G1:**

**When count will be 1:**

```
FOR new_count ← 1 to 6
    INPUT new_item_code[1]
NEXT new_count
```

**Input will be:**

D2

**When count will be 2:**

```
    INPUT new_item_code[2]
```

**Input will be:**

E1

**When count will be 3:**

```
    INPUT new_item_code[3]
```

**Input will be:**

E2

**When count will be 4:**

```
    INPUT new_item_code[4]
```

**Input will be:**

E3

**When count will be 5:**

```
    INPUT new_item_code[5]
```

**Input will be:**

F2

**When count will be 6:**

```
    INPUT new_item_code[6]
```

**Input will be:**

G1

**The array of new_item_code will be updated as the loops keeps incrementing and new_count increases by 1 every time loop is repeated**

| *new_count* | new_item_code |
|---|---|
| 1 | D2 |
| 2 | E1 |
| 3 | E2 |
| 4 | E3 |
| 5 | F2 |
| 6 | G1 |

- update the price of the computer
- store and output the additional items and the new price of the computer.

First we have to store the details of additional items in separate arrays (not variables unlike **TASK 1** because there was fixed number of items to be chosen: 1 case, 1 RAM and 1 Main HDD. But in **TASK 2** there is no fixed number as the customer can choose either 1, 2, 3, 4, 5, 6, 7, 8, 9 or even more additional items) and then accordingly recalculate the price of the computer using the cost of those additional items. We will add the previously calculated price and cost of newly bought additional items to work out the new total price.

We will use the **same method used in TASK 1 to store the details** of additional chosen items with their prices.

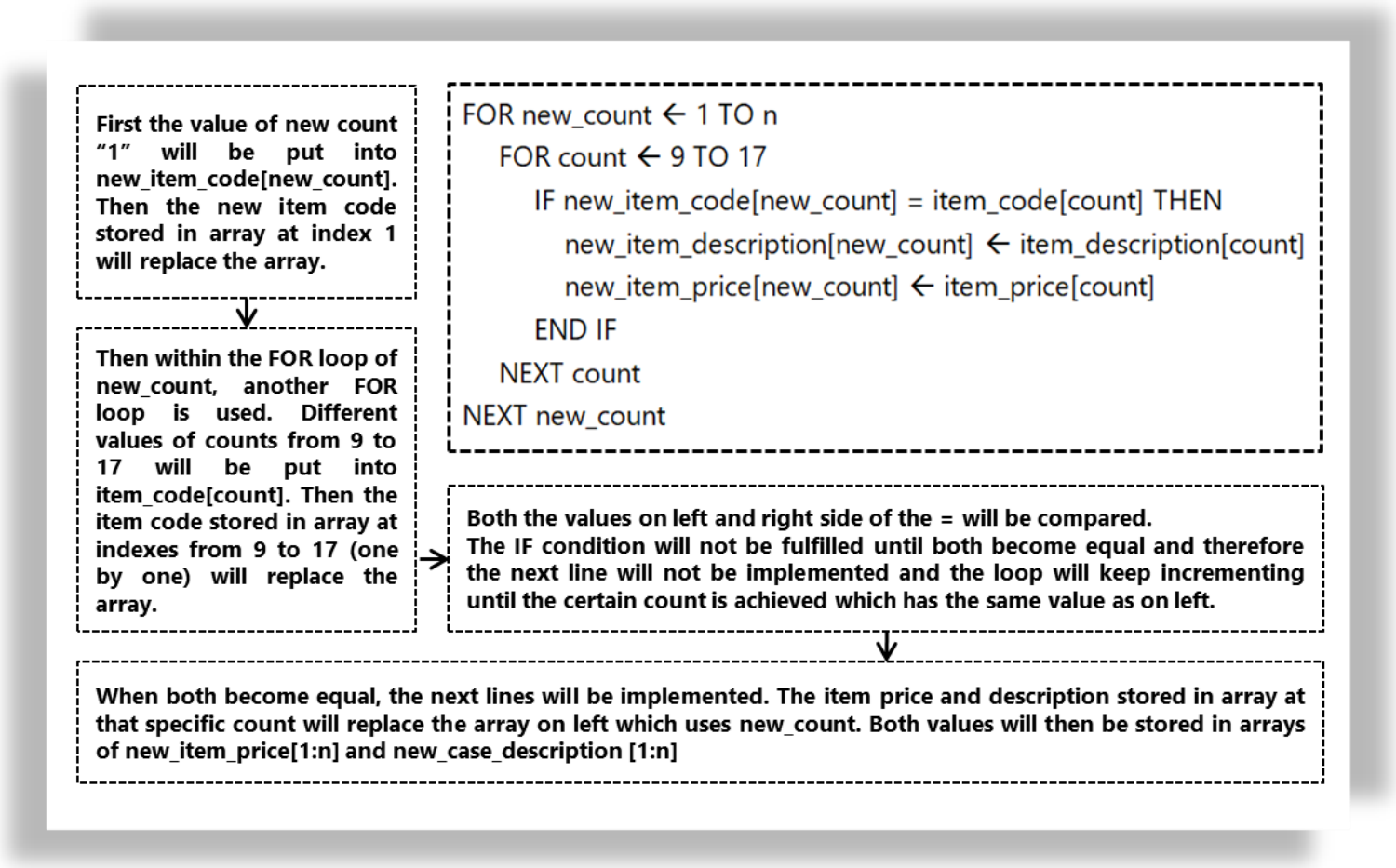
**1) The method is to store the details of every additional chosen item code in their arrays using nested *FOR...TO...NEXT* loop:**

We will use one ***FOR*** loop and another ***nested FOR*** loop as well. Outer loop will have a ***new_count*** from 1 to the number of additional items bought ***(n)***. The inner loop will have a count from 9 to 17 so that it runs for all the additional 9 items which are possible choices for the customer.

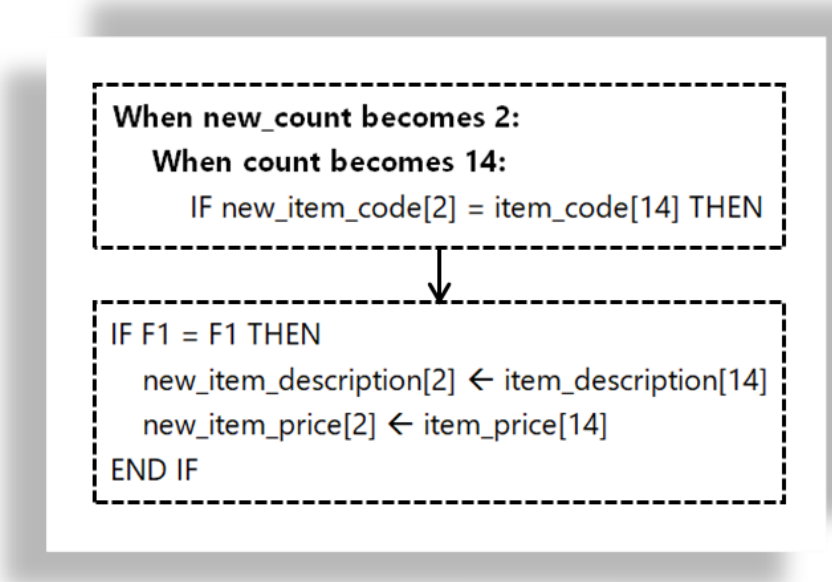
```
FOR new_count ← 1 TO n
    FOR count ← 9 TO 17
        IF new_item_code[new_count] = item_code[count] THEN
            new_item_description[new_count] ← item_description[count]
            new_item_price[new_count] ← item_price[count]
        END IF
    NEXT count
NEXT new_count
```

Now this may seem confusing at first but after understanding the running example of this code and the process that is being carried out, it will seem the most efficient and time saving method.

**Running of example code:**




Suppose a customer chose to buy 3 additional items. The 2nd additional item chosen by the customer is F1. Now when the ***new_count*** will become 2, the ***nested FOR*** loop will run from 9 to 17. The loop will keep incrementing until the details of all the additional items are stored. If both values do not match then the count will keep **increasing (+1)** until ultimately any one value will match and its details will be stored. In this case, when the count of ***nested FOR*** loop will become 14 then it will look somehow like this:




**Since both values become equal, the item description stored for F1 in index 14 will be stored in new item description in index 2 as well. The same goes for item price.**

- update the price of the computer

Now we have stored the details of additional chosen items along with their prices. The final step is to recalculate and update the **total price** of the computer and then output details with **new total price**.

We will use ***FOR*** loop and ***concept of totaling***. The total for new items will keep updating with every single repetition of ***FOR*** loop.

```
FOR new_count ← 1 TO n
    new_items_total_price ← new_items_total_price + new_item_price[new_count]
NEXT new_count
```

The final step is to simply add the **total for new items** with the **previously calculated final price** and hence obtain the **new total**.

```
new_total_price ← total_price + new_items_total_price
```

- store and output the additional items and the new price of the computer.

In the end, we will finally **PRINT the required output** only (as we have already stored the details of additional items and new price of computer).

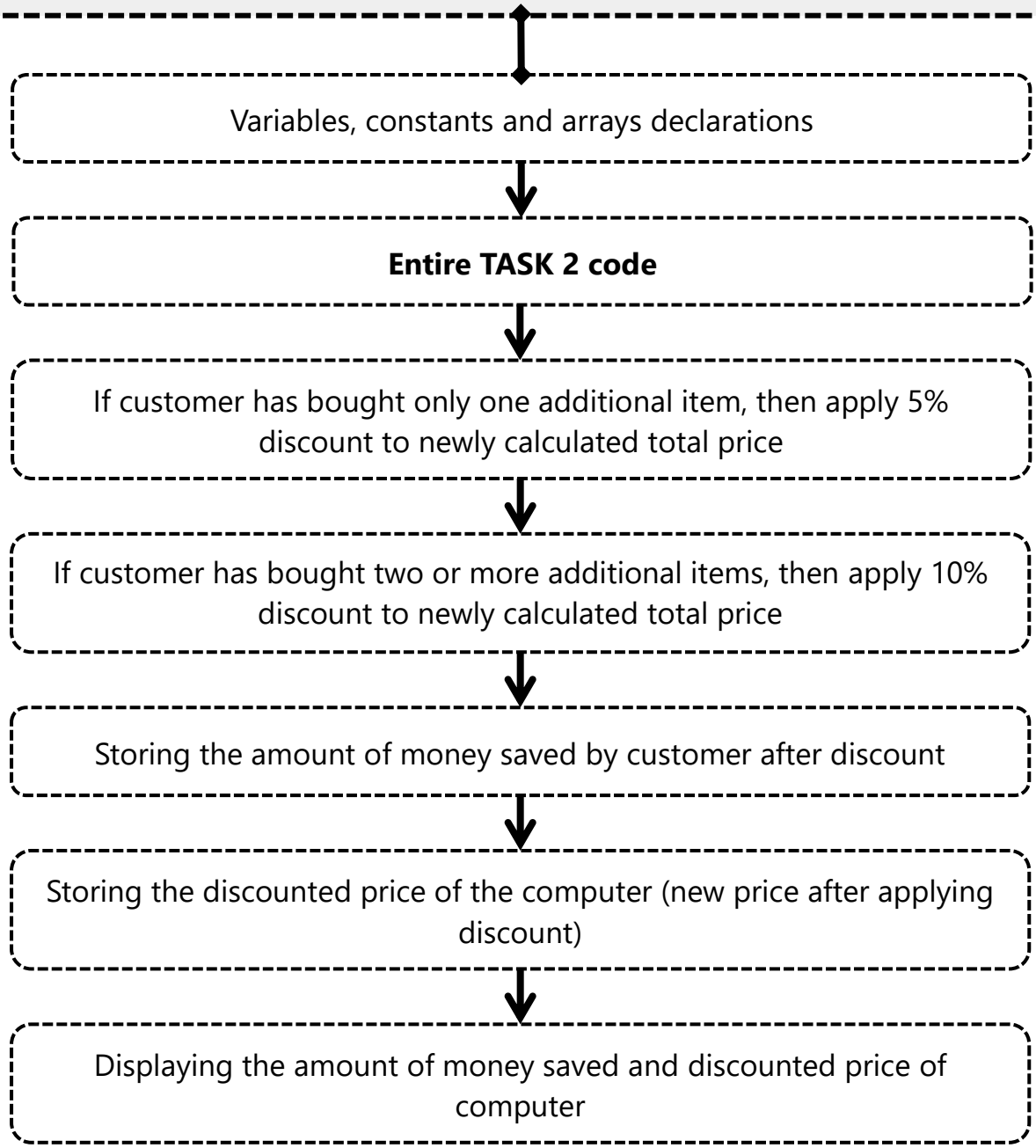
If the customer does not wish to buy additional items then we will simply **PRINT an appropriate message** thanking the customer for shopping.

## Concept and understanding of TASK 3:

**Task 3** – Offering discounts.

Extend TASK 2 to:

- apply a 5% discount to the price of the computer if the customer has bought only one additional item
- apply a 10% discount to the price of the computer if the customer has bought two or more additional items
- output the amount of money saved and the new price of the computer after the discount.



## Explanation of Algorithm of TASK 3:

In this task, we have to give either a **5% or 10% discount** depending upon if and how many additional items the customer bought and then calculating the **discounted total price** of the computer and storing the **amount of money saved** due to discount.

Extend TASK 2 to:

This means that the **identifiers and Pseudocode used in TASK 2** will be used for TASK 3 just like they are but with few extensions. TASK 3 will simply **extend** the conditions/demands of **TASK 2** and then finish.

- Apply a 5% discount to the price of the computer if the customer has bought only one additional item

- Apply a 10% discount to the price of the computer if the customer has bought two or more additional items

In **TASK 2**, the customer gave an input ***(n)*** which stored the number of additional items the customer wants to buy. Now if the customer gave the value 3 then he/she bought 3 additional items. If the customer had given the value 8 then he/she bought 8 additional items.

The basic idea of explaining this variable ***(n)*** again is that now we have to give customer discounts based upon this value:

- If the customer bought only one additional item (meaning ***n* = 1**) then a **5% discount** will be given.
- If the customer bought more than one additional item (meaning ***n* > 1)** then a **10% discount** will be given.

We will determine the number of additional items customer bought using the ***IF*** selection statement. Then accordingly we will store the amount of discount as well.

Someone with a basic knowledge of percentages and mathematics can make sense of the following calculation:

$\frac{5}{100} = 0.05$ Multiplying this 0.05 with the new total price calculated in TASK 2 will give us the amount of money the customer will save on 5% discount.

$\frac{10}{100} = 0.1$ Multiplying this 0.1 with the new total price calculated in TASK 2 will give us the amount of money the customer will save on 10% discount.

The **amount of money saved** will be stored in a variable. The use of ***IF*** selection statement together with these simple percentage calculations becomes the following piece of code:

```
IF n = 1 THEN
    discount ← 0.05
ELSE IF n > 1 THEN
    discount ← 0.1
END IF
money_saved ← new_total_price * discount
```

Then in another calculation we will simply subtract the **saved amount** from the **new total price** calculated in **TASK 2**. This will give us the total price after giving discount called **discounted price**.

```
discounted_price ← new_total_price – money_saved
```

**Another method is to:**

- using the ***IF*** selection statement to determine the number of additional items and storing either 0.95 or 0.9 as the amount of discount.
- multiplying 0.95 with the **new total price** calculated in **TASK 2** will give us the amount of total money after a **5%** discount called **discounted price**.
- multiplying 0.90 with the **new total price** calculated in **TASK 2** will give us the amount of total money after a **10%** discount called **discounted price**.
- subtracting the **new total price** calculated in **TASK 2** from **discounted price** will give us the amount of **money saved**.

- output the amount of money saved and the new price of the computer after the discount.

In the end, we will finally **PRINT the required output** (e.g. ***money_saved*** and ***discounted_price***).

## Color Codes:

The Pseudo codes for **TASK 1, TASK 2** and **TASK 3** uses different colors for representing some keywords for a better understanding. These keywords and their color codes are listed below in two tables below:

| | |
|---|---|
| **Begin & End** | **BLACK** |
| **Declarations & Data Types** | **LIGHT BLUE** |
| **Input & Output** | **DARK BLUE** |
| **Pre-defined Values of Arrays** | **ORANGE** |
| **Prompts, Messages & Variables** | **BLACK** |

| | |
|---|---|
| **Logical, Arithmetic & Relational Operators** | **GREEN** |
| **Selection Statements (IF…THEN…END IF)** | **RED** |
| **Loop (FOR…TO…NEXT)** | **PURPLE** |
| **Loop (WHILE…DO…END WHILE)** | **PINK** |

# TASK 1 – Pseudocode:

**BEGIN**

```
CONST initial_price ← 200.0 AS FLOAT
DECLARE count ← 0 AS INTEGER
DECLARE case_code ← "", ram_code ← "", main_disk_code ← "" AS STRING
DECLARE case_description ← "", ram_description ← "", main_disk_description ← "" AS STRING
DECLARE item_code [1:17], item_description [1:17] AS STRING
DECLARE case_price ← 0.0, ram_price ← 0.0, main_disk_price ← 0.0, total_price ← 0.0 AS FLOAT
DECLARE item_price [1:17] AS FLOAT

item_code[1] ← "A1"
item_code[2] ← "A2"
item_code[3] ← "B1"
item_code[4] ← "B2"
item_code[5] ← "B3"
item_code[6] ← "C1"
item_code[7] ← "C2"
item_code[8] ← "C3"
item_code[9] ← "D1"
item_code[10] ← "D2"
item_code[11] ← "E1"
item_code[12] ← "E2"
item_code[13] ← "E3"
item_code[14] ← "F1"
item_code[15] ← "F2"
item_code[16] ← "G1"
item_code[17] ← "G2"

item_description[1] ← "Case Compact"
item_description[2] ← "Case Tower"
item_description[3] ← "RAM 8 GB"
item_description[4] ← "RAM 16 GB"
item_description[5] ← "RAM 32 GB"
item_description[6] ← "Main Hard Disk Drive 1 TB HDD"
item_description[7] ← "Main Hard Disk Drive 2 TB HDD"
item_description[8] ← "Main Hard Disk Drive 4 TB HDD"
item_description[9] ← "Solid State Drive 240 GB SSD"
item_description[10] ← "Solid State Drive 480 GB SSD"
item_description[11] ← "Second Hard Disk Drive 1 TB HDD"
item_description[12] ← "Second Hard Disk Drive 2 TB HDD"
```

```
item_description[13] ← "Second Hard Disk Drive 4 TB HDD
item_description[14] ← "Optical Drive DVD/Blu-Ray Player"
item_description[15] ← "Optical Drive DVD/Blu-Ray Re-writer"
item_description[16] ← "Operating System Standard Version"
item_description[17] ← "Operating System Professional Version"

item_price[1] ← 75.00
item_price[2] ← 150.00
item_price[3] ← 79.99
item_price[4] ← 149.99
item_price[5] ← 299.99
item_price[6] ← 49.99
item_price[7] ← 89.99
item_price[8] ← 129.99
item_price[9] ← 59.99
item_price[10] ← 119.99
item_price[11] ← 49.99
item_price[12] ← 89.99
item_price[13] ← 129.99
item_price[14] ← 50.00
item_price[15] ← 100.00
item_price[16] ← 100.00
item_price[17] ← 175.00

PRINT "New sale initiated - Default basic set of components costing $200 is added."
PRINT "One case, one RAM AND one Main Hard Disk Drive is required to be added."

PRINT “The following are the case item codes along with their descriptions and prices:”

FOR count ← 1 TO 2
  PRINT “Item code: “, item_code[count]
  PRINT “Item description: “, item_description[count]
  PRINT “Item price: “, item_price[count]
NEXT count

PRINT "Enter the case item code (A1 or A2)."
        INPUT case_code
WHILE case_code <> “A1” OR case_code <> “A2”
        INPUT “Wrong case item code. It should be either A1 or A2 only.”, case_code
END WHILE
```

```
PRINT "The following are the RAM item codes along with their descriptions and prices:"
FOR count ← 3 TO 5
  PRINT "Item code: ", item_code[count]
  PRINT "Item description: ", item_description[count]
  PRINT "Item price: ", item_price[count]
NEXT count

PRINT "Enter the RAM item code (B1 or B2 or B3)."
      INPUT ram_code
WHILE ram_code <> "B1" OR ram_code <> "B2" OR ram_code <> "B3"
      INPUT "Wrong RAM item code. It should be either B1, B2 or B3 only", ram_code
END WHILE

PRINT "The following are the Main Hard Disk Drive item codes along with their descriptions and prices: "
FOR count ← 6 TO 8
  PRINT "Item code: ", item_code[count]
  PRINT "Item description: ", item_description[count]
  PRINT "Item price: ", item_price[count]
NEXT count

PRINT "Enter the Main Hard Disk Drive item code (C1 or C2 or C3)."
      INPUT main_disk_code
WHILE main_disk_code <> "C1" OR main_disk_code <> "C2" OR main_disk_code <> "C3"
      INPUT "Wrong Main Hard Disk Drive item code. It should be either C1, C2 or C3 only",
      main_disk_code
END WHILE

FOR count ← 1 TO 8
      IF item_code[count] = case_code THEN
         case_price ← item_price[count], case_description ← item_description[count]
      END IF

      IF item_code[count] = ram_code THEN
         ram_price ← item_price[count], ram_description ← item_description[count]
      END IF

      IF item_code[count] = main_disk_code THEN
         main_disk_price ← item_price[count], main_disk_description ← item_description[count]
      END IF
NEXT count
```

```
    total_price ← initial_price + case_price + ram_price + main_disk_price

    PRINT "The case, RAM and Main Hard Disk Drive bought along with their codes, descriptions and prices are:"
    PRINT case_code, case_description, case_price
    PRINT ram_code, ram_description, ram_price
    PRINT main_disk_code, main_disk_description, main_disk_price
    PRINT "The total price of the computer after buying the required items is: $", total_price
END
```

## TASK 1 – Efficiency:

- Use of ***CONSTANT*** to hold fixed value of items.
- Use of ***ARRAY*** to store item code, description and price.
- Initialization of all ***ARRAYS*** with pre-defined values.
- Use of ***FOR*** loops to output details of items.
- Use of ***WHILE*** loop to validate all user inputs and output appropriate error messages when validation fails.
- Use of ***IF*** statement to determine the items chosen by customer.

# TASK 1 – Explanation of Pseudocode:

```
CONST initial_price ← 200.0 AS FLOAT
DECLARE count ← 0 AS INTEGER
DECLARE case_code ← "", ram_code ← "", main_disk_code ← "" AS STRING
DECLARE case_description ← "", ram_description ← "", main_disk_description ← "" AS STRING
DECLARE item_code [1:17], item_description [1:17] AS STRING
DECLARE case_price ← 0.0, ram_price ← 0.0, main_disk_price ← 0.0, total_price ← 0.0 AS FLOAT
DECLARE item_price [1:17] AS FLOAT
```

Declaration of variables, constants and arrays.

```
item_code[1] ← "A1"
item_code[2] ← "A2"
item_code[3] ← "B1"
item_code[4] ← "B2"
item_code[5] ← "B3"
item_code[6] ← "C1"
item_code[7] ← "C2"
item_code[8] ← "C3"
item_code[9] ← "D1"
item_code[10] ← "D2"
item_code[11] ← "E1"
item_code[12] ← "E2"
item_code[13] ← "E3"
item_code[14] ← "F1"
item_code[15] ← "F2"
item_code[16] ← "G1"
item_code[17] ← "G2"

item_description[1] ← "Case Compact"
item_description[2] ← "Case Tower"
item_description[3] ← "RAM 8 GB"
item_description[4] ← "RAM 16 GB"
item_description[5] ← "RAM 32 GB"
item_description[6] ← "Main Hard Disk Drive 1 TB HDD"
item_description[7] ← "Main Hard Disk Drive 2 TB HDD"
item_description[8] ← "Main Hard Disk Drive 4 TB HDD"
item_description[9] ← "Solid State Drive 240 GB SSD"
item_description[10] ← "Solid State Drive 480 GB SSD"
item_description[11] ← "Second Hard Disk Drive 1 TB HDD"
item_description[12] ← "Second Hard Disk Drive 2 TB HDD"
item_description[13] ← "Second Hard Disk Drive 4 TB HDD
item_description[14] ← "Optical Drive DVD/Blu-Ray Player"
item_description[15] ← "Optical Drive DVD/Blu-Ray Re-writer"
item_description[16] ← "Operating System Standard Version"
item_description[17] ← "Operating System Professional Version"

item_price[1] ← 75.00
item_price[2] ← 150.00
item_price[3] ← 79.99
item_price[4] ← 149.99
item_price[5] ← 299.99
item_price[6] ← 49.99
item_price[7] ← 89.99
item_price[8] ← 129.99
item_price[9] ← 59.99
```

Assigning pre-defined values to 3 different arrays.

```
item_price[10] ← 119.99
item_price[11] ← 49.99
item_price[12] ← 89.99
item_price[13] ← 129.99
item_price[14] ← 50.00
item_price[15] ← 100.00
item_price[16] ← 100.00
item_price[17] ← 175.00
```

```
PRINT "New sale initiated - Default basic set of components costing $200 is added."
PRINT "One case, one RAM AND one Main Hard Disk Drive is required to be added."

PRINT "The following are the case item codes along with their descriptions and prices:"
FOR count ← 1 TO 2
  PRINT "Item code: ", item_code[count]
  PRINT "Item description: ", item_description[count]
  PRINT "Item price: ", item_price[count]
NEXT count
```

Use of FOR loop to output details of two cases.

```
PRINT "Enter the case item code (A1 or A2)."
      INPUT case_code
WHILE case_code <> "A1" OR case_code <> "A2"
      INPUT "Wrong case item code. It should be either A1 or A2 only.", case_code
END WHILE
```

Input and validation of one chosen case using WHILE loop.

```
PRINT "The following are the RAM item codes along with their descriptions and prices:"
FOR count ← 3 TO 5
  PRINT "Item code: ", item_code[count]
  PRINT "Item description: ", item_description[count]
  PRINT "Item price: ", item_price[count]
NEXT count
```

Use of FOR loop to output details of three RAM's.

```
PRINT "Enter the RAM item code (B1 or B2 or B3)."
      INPUT ram_code
WHILE ram_code <> "B1" OR ram_code <> "B2" OR ram_code <> "B3"
      INPUT "Wrong RAM item code. It should be either B1, B2 or B3 only", ram_code
END WHILE
```

Input and validation of one chosen RAM using WHILE loop.

```
PRINT "The following are the Main Hard Disk Drive item codes along with their descriptions and prices: "
FOR count ← 6 TO 8
  PRINT "Item code: ", item_code[count]
  PRINT "Item description: ", item_description[count]
  PRINT "Item price: ", item_price[count]
NEXT count
```

Use of FOR loop to output details of three Main Hard Disk Drives.

```
PRINT "Enter the Main Hard Disk Drive item code (C1 or C2 or C3)."
      INPUT main_disk_code
WHILE main_disk_code <> "C1" OR main_disk_code <> "C2" OR main_disk_code <> "C3"
      INPUT "Wrong Main Hard Disk Drive item code. It should be either C1, C2 or C3 only",
      main_disk_code
END WHILE
```

Input and validation of one chosen Main HDD using WHILE loop.

```
FOR count ← 1 TO 8
    IF item_code[count] = case_code THEN
        case_price ← item_price[count], case_description ← item_description[count]
    END IF

    IF item_code[count] = ram_code THEN
        ram_price ← item_price[count], ram_description ← item_description[count]
    END IF

    IF item_code[count] = main_disk_code THEN
        main_disk_price ← item_price[count], main_disk_description ← item_description[count]
    END IF
NEXT count
```

Use of FOR loop to check every single item whether it was chosen or not by the customer for all 8 items.

Use of IF selection statement to determine which case, RAM and Main HDD was chosen by the customer out of total 8 items option and then storing their details separately.

```
total_price ← initial_price + case_price + ram_price + main_disk_price
```

Calculation of the total price of the computer.

```
PRINT "The case, RAM and Main Hard Disk Drive bought along with their codes, descriptions and prices are:"
PRINT case_code, case_description, case_price
PRINT ram_code, ram_description, ram_price
PRINT main_disk_code, main_disk_description, main_disk_price
PRINT "The total price of the computer after buying the required items is: $", total_price
```

Required output of TASK 1.

# TASK 1 – Expected Questions:

1. State two variables you used for Task 1. State the data type and purpose of the variable.
2. State three arrays you used for Task 1. State the data type and purpose of the arrays.
3. Describe the data structures you have used in Task 1 to store the data for the computer components. Include the name(s), data type, sample data and usage for each structure.
4. Write an algorithm for Task 1, using either Pseudocode, programming statements or a flowchart. Assume that the data structures for storing data about computer components have already been initialized with predefined values.
5. Write an algorithm to complete Task 1 without including any error prompts, using either Pseudocode, programming statements or a flowchart.
6. Explain how your program completes/performs Task 1. Any programming statements used in your answer must be fully explained.
7. Explain how you allowed a customer to only choose one case, one RAM and one Main Hard Disk Drive from all the items (part of Task 1)?
8. Explain how you calculated the total price of the computer (part of Task 1)? You can include Pseudocode or programming statements as part of your explanation.
9. Explain how you stored the details of chosen items with their prices (part of Task 1)? You can include Pseudocode or programming statements as part of your explanation.
10. Explain how you validated any two inputs used in Task 1. State one valid and one invalid input to test your validation methods (valid and invalid test data). You can include Pseudocode or programming statements as part of your explanation.
11. Write an algorithm for Task 1, using either Pseudocode, programming statements or a flowchart. Change the algorithm to ensure that the customer must buy an operating system as well with the case, RAM and Main Hard Disk Drive. Assume that the data structures for storing data about computer components have already been initialized with predefined values.
12. The cost of basic set of components in computer has been changed to $300 instead of $200. Explain the changes you would make to your program to calculate the total price of the computer? You can include Pseudocode or programming statements as part of your explanation.
13. Comment on the efficiency of your code for Task 1.

# TASK 2 – Pseudocode:

**BEGIN**

**[ALL IDENTIFIERS OF TASK 1]**

```
DECLARE n ← 0, new_count ← 0 AS INTEGER
DECLARE new_item_code [1:n], new_item_description [1:n] AS STRING
DECLARE new_items_total_price ← 0.0, new_total_price ← 0.0 AS FLOAT
DECLARE new_item_price [1:n] AS FLOAT
DECLARE choice AS CHAR
```

**[ENTIRE PSEUDOCODE OF TASK 1]**

```
PRINT "Do you want to buy any additional items? (Y or N)."
   INPUT choice
WHILE choice <> "Y" OR choice <> "N"
   INPUT "Wrong input. It should be either Y or N only.", choice
END WHILE

IF choice = "Y" THEN
   PRINT "The following are the additional item codes along with their descriptions and prices:"
   FOR count ← 9 TO 17
     PRINT "Item code: ", item_code[count]
     PRINT "Item description: ", item_description[count]
     PRINT "Item price: ", item_price[count]
   NEXT count

   PRINT "How many additional items do you want to buy?"
      INPUT n
   WHILE n < 1
      INPUT "You cannot buy less than atleast 1 additional item(s).", n
   END WHILE

   FOR new_count ← 1 TO n
      PRINT "Enter the additional item(s) code."
         INPUT new_item_code[new_count]
   WHILE new_item_code <> "D1" OR new_item_code <> "D2" OR new_item_code <> "E1" OR
   new_item_code <> "E2" OR new_item_code <> "E3" OR new_item_code <> "F1" OR new_item_code
   <> "F2" OR new_item_code <> "G1" OR new_item_code <> "G2"
         INPUT "Wrong additional item code. It should be either D1, D2, E1, E2, E3, F1, F2, G1, or G2
         only.", new_item_code[new_count]
   END WHILE
   NEXT new_count
```

```
    FOR new_count ← 1 TO n
        FOR count ← 9 TO 17
            IF new_item_code[new_count] = item_code[count] THEN
                new_item_description[new_count] ← item_description[count]
                new_item_price[new_count] ← item_price[count]
            END IF
        NEXT count
    NEXT new_count

    FOR new_count ← 1 TO n
        new_items_total_price ← new_items_total_price + new_item_price[new_count]
    NEXT new_count

    new_total_price ← total_price + new_items_total_price

    PRINT "The additional items bought along with their codes, descriptions and prices are:"
    FOR new_count ← 1 TO n
        PRINT new_item_code[new_count], new_item_description[new_count], new_item_price[new_count]
    NEXT new_count
    PRINT "The total price of the computer, items bought previously and new items is: $", new_total_price
  END IF

  IF choice = "N" THEN
    PRINT "Thank you for shopping."
  END IF
END
```

**ENTIRE EFFICIENCY OF TASK 1 can be written as well.**

## TASK 2 – Efficiency:

- Use of ***ARRAY*** to store additional item code, description and price.
- Use of ***FOR*** loops to output details of additional items and then input additional item codes.
- Use of ***FOR*** loops to total the new items price and then output details of items bought by customer.
- Use of ***WHILE*** loop to validate all user inputs and output appropriate error messages when validation fails.
- Use of ***IF*** statement to determine the additional items chosen by customer and to determine customers choice.

# TASK 2 – Explanation of Pseudocode:

**[ALL IDENTIFIERS OF TASK 1]**

```
DECLARE n ← 0, new_count ← 0 AS INTEGER
DECLARE new_item_code [1:n], new_item_description [1:n] AS STRING
DECLARE new_items_total_price ← 0.0, new_total_price ← 0.0 AS FLOAT
DECLARE new_item_price [1:n] AS FLOAT
DECLARE choice AS CHAR
```

Declaration of variables, constants and arrays (including TASK 1).

**[ENTIRE PSEUDOCODE OF TASK 1]**

**Complete TASK 1 (without any change).**

```
PRINT "Do you want to buy any additional items? (Y or N)."
  INPUT choice
WHILE choice <> "Y" OR choice <> "N"
  INPUT "Wrong input. It should be either Y or N only.", choice
END WHILE
```

Input and validation of customers choice about whether to buy any additional items or not.

```
IF choice = "Y" THEN
```

Use of IF selection statement to determine customers choice about purchasing additional items.

```
  PRINT "The following are the additional item codes along with their descriptions and prices:"
  FOR count ← 9 TO 17
    PRINT "Item code: ", item_code[count]
    PRINT "Item description: ", item_description[count]
    PRINT "Item price: ", item_price[count]
  NEXT count
```

Use of FOR loop to output details of remaining 9 additional items.

```
  PRINT "How many additional items do you want to buy?"
    INPUT n
  WHILE n < 1
    INPUT "You cannot buy less than atleast 1 additional item(s).", n
  END WHILE
```

Input and validation of customers choice about how many additional items to buy using WHILE loop.

```
  FOR new_count ← 1 TO n
    PRINT "Enter the additional item(s) code."
      INPUT new_item_code[new_count]
    WHILE new_item_code <> "D1" OR new_item_code <> "D2" OR new_item_code <> "E1" OR
    new_item_code <> "E2" OR new_item_code <> "E3" OR new_item_code <> "F1" OR new_item_code
    <> "F2" OR new_item_code <> "G1" OR new_item_code <> "G2"
      INPUT "Wrong additional item code. It should be either D1, D2, E1, E2, E3, F1, F2, G1, or G2
      only.", new_item_code[new_count]
    END WHILE
  NEXT new_count
```

Use of FOR loop to input the codes of additional item(s).

Input and validation of additional chosen items using WHILE loop.

```
  FOR new_count ← 1 TO n
    FOR count ← 9 TO 17
      IF new_item_code[new_count] = item_code[count] THEN
        new_item_description[new_count] ← item_description[count]
        new_item_price[new_count] ← item_price[count]
      END IF
    NEXT count
  NEXT new_count
```

- Use of outer FOR loop to check which additional items were bought by the customer.
- Use of inner FOR loop to check every single choice of additional item whether it was chosen or not by the customer for remaining 9 items.

Use of IF selection statement to determine which additional items were chosen by the customer and then storing their details separately using arrays.

```
FOR new_count ← 1 TO n
    new_items_total_price ← new_items_total_price + new_item_price[new_count]
NEXT new_count
```

Use of FOR loop to calculate the total price of new items.

```
new_total_price ← total_price + new_items_total_price
```

Calculation of the new total price of the computer.

```
PRINT "The additional items bought along with their codes, descriptions and prices are:"
FOR new_count ← 1 TO n
    PRINT new_item_code[new_count], new_item_description[new_count], new_item_price[new_count]
NEXT new_count
PRINT "The total price of the computer, items bought previously and new items is: $", new_total_price
END IF
```

Use of FOR loop to output additional item(s) details.

Required output of TASK 2.

```
IF choice = "N" THEN
    PRINT "Thank you for shopping."
END IF
```

Use of IF selection statement to determine customers choice about purchasing additional items and if not then output an appropriate message.

# TASK 2 – Expected Questions:

1. State two variables you used for Task 2. State the data type and purpose of the variable.
2. State three arrays you used for Task 2. State the data type and purpose of the arrays.
3. Describe the data structures you have used in Task 2 to store the data for the additional computer components. Include the name(s), data type, sample data and usage for each structure.
4. Write an algorithm for Task 2, using either Pseudocode, programming statements or a flowchart. You should assume that Task 1 has already been completed.
5. Write an algorithm to complete Task 2 without including any error prompts, using either Pseudocode, programming statements or a flowchart. You should assume that Task 1 has already been completed.
6. Explain how your program completes/performs Task 2. Any programming statements used in your answer must be fully explained.
7. Explain how Task 1 has been extended to meet the requirements for Task 2. Any programming statements used in your answer must be fully explained.
8. Explain how you calculated and updated the total price of the computer (part of Task 2)? You can include Pseudocode or programming statements as part of your explanation.
9. Explain how you stored the details of additional chosen items with their prices (part of Task 2)? You can include Pseudocode or programming statements as part of your explanation.
10. Explain how you validated any two inputs used in Task 2. State one valid and one invalid input to test your validation methods (valid and invalid test data). You can include Pseudocode or programming statements as part of your explanation.
11. Write an algorithm for Task 2, using either Pseudocode, programming statements or a flowchart. Change the algorithm to ensure that the customer cannot buy more than 3 additional items. You should assume that Task 1 has already been completed.
12. Comment on the efficiency of your code for Task 2.

# TASK 3 – Pseudocode:

```
BEGIN

    [ALL IDENTIFIERS OF TASK 2]

    DECLARE discount ← 0.0, money_saved ← 0.0, discounted_price ← 0.0 AS FLOAT

    [ENTIRE PSEUDOCODE OF TASK 2]

    IF n = 1 THEN
        discount ← 0.05
        PRINT "You have bought only one additional item. You would be given a 5% discount!"
    ELSE IF n > 1 THEN
        discount ← 0.1
        PRINT "You have bought more than one additional items. You would be given a 10% discount!"
    END IF

    money_saved ← new_total_price * discount
    discounted_price ← new_total_price – money_saved

    PRINT "The amount of money saved is: $", money_saved
    PRINT "The new price of the computer after discount is: $", discounted_price

END
```

**ENTIRE EFFICIENCY OF TASK 2 can be written as well.**

# TASK 3 – Efficiency:

- Use of ***IF*** statement to determine the discount that customer will be given.
- ***CALCULATION*** of money saved and discounted price using a ***VARIABLE*** from TASK 2.
- Giving the required ***OUTPUT*** with appropriate/proper messages.

# TASK 3 – Explanation of Pseudocode:

**[ALL IDENTIFIERS OF TASK 2]**

```
DECLARE discount ← 0.0, money_saved ← 0.0, discounted_price ← 0.0 AS FLOAT
```

Declaration of variables, constants and arrays (including TASK 2).

**[ENTIRE PSEUDOCODE OF TASK 2]** — **Complete TASK 2 (without any change).**

```
IF n = 1 THEN
    discount ← 0.05
    PRINT "You have bought only one additional item. You would be given a 5% discount!"
ELSE IF n > 1 THEN
    discount ← 0.1
    PRINT "You have bought more than one additional items. You would be given a 10% discount!"
END IF
```

Use of IF selection statement to determine the amount of discount customer will be given.

```
money_saved ← new_total_price * discount
discounted_price ← new_total_price – money_saved
```

Calculation of the money saved and discounted price.

```
PRINT "The amount of money saved due to discount is: $", money_saved
PRINT "The new price of the computer after discount is: $", discounted_price
```

Required output of TASK 3.

# TASK 3 – Expected Questions:

1. State two variables you used for Task 3. State the data type and purpose of the variable.
2. Write an algorithm for Task 3, using either Pseudocode, programming statements or a flowchart. You should assume that Task 1 and Task 2 have already been completed.
3. Explain how your program completes/performs Task 3. Any programming statements used in your answer must be fully explained.
4. Explain how Task 2 has been extended to meet the requirements for Task 3. Any programming statements used in your answer must be fully explained.
5. Explain how you calculated the discount on the overall price of the computer in Task 3. You can include Pseudocode or programming statements as part of your explanation.
6. Explain how you calculated the money saved by the customer after discount on the overall price of the computer (part of Task 3). You can include Pseudocode or programming statements as part of your explanation.
7. Write an algorithm for Task 3, using either Pseudocode, programming statements or a flowchart. Change the algorithm so that if discount is applicable, then the program outputs the amount of money saved as a percentage of the initial price of the computer. You should assume that Task 1 and Task 2 have already been completed.
8. The discount applicable for customer on buying two or three additional items is 15% and buying 4 or more additional items is 30%. Explain the changes you would make to your program to calculate this discount on price of computer? You can include Pseudocode or programming statements as part of your explanation.
9. Comment on the efficiency of your code for Task 3.