PRE-RELEASE MATERIAL

May/June 2019

O Level Computer Science 2210/22

Abstract

This document provides solution to the Cambridge pre-release material for Computer Science 2210

/blitzcomputing

info@blitz.education

Prepared by: Blitz Computing

Pre-Release Material Tasks

An auction company has an interactive auction board at their sale rooms, which allows buyers to place bids at any time during the auction. Before the auction starts, the sellers place their items in the sale room with a unique number attached to each item (item number). The following details about each item need to be set up on the interactive auction board system: item number, number of bids, description and reserve price. The number of bids is initially set to zero.

During the auction, buyers can look at the items in the sale room and then place a bid on the interactive auction board at the sale room. Each buyer is given a unique number for identification (buyer number). All the buyer needs to do is enter their buyer number, the item number and their bid. Their bid must be greater than any existing bids.

At the end of the auction, the company checks all the items and marks those that have bids greater than the reserve as sold. Any items sold will incur a fee of 10% of the final bid to be paid to the auction company.

Write and test a program or programs for the auction company.

- 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 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 – Auction set up

For every item in the auction the item number, description and the reserve price should be recorded. The number of bids is set to zero. There must be at least 10 items in the auction.

TASK 2 – Buyer bids

A buyer should be able to find an item and view the item number, description and the current highest bid. A buyer can then enter their buyer number and bid, which must be higher than any previously recorded bids. Every time a new bid is recorded the number of bids for that item is increased by one. Buyers can bid for an item many times and they can bid for many items.

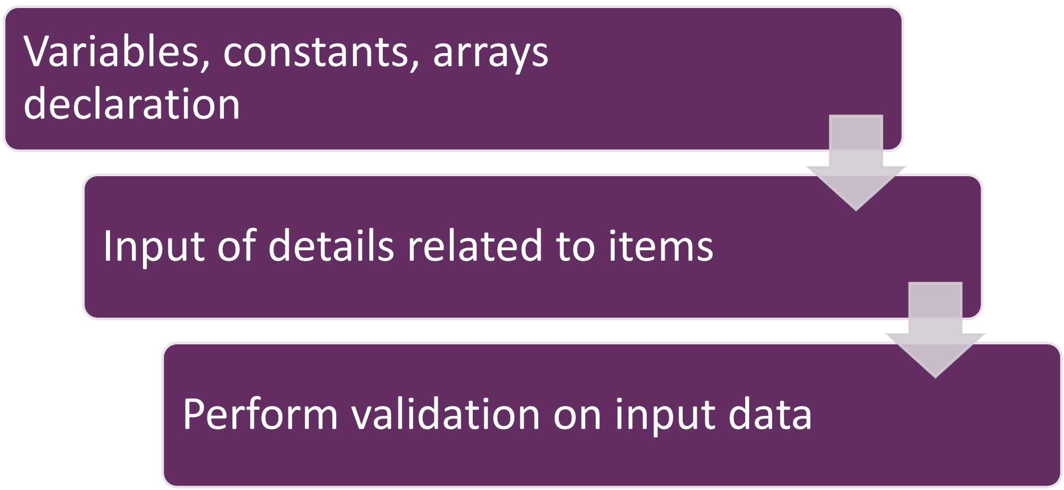
TASK 3 – At the end of the auction

Using the results from TASK 2, identify items that have reached their reserve price, mark them as sold, calculate 10% of the final bid as the auction company fee and add this to the total fee for all sold items. Display this total fee. Display the item number and final bid for all the items with bids that have not reached their reserve price. Display the item number of any items that have received no bids. Display the number of items sold, the number of items that did not meet the reserve price and the number of items with no bids.

Concept and Understanding of Tasks

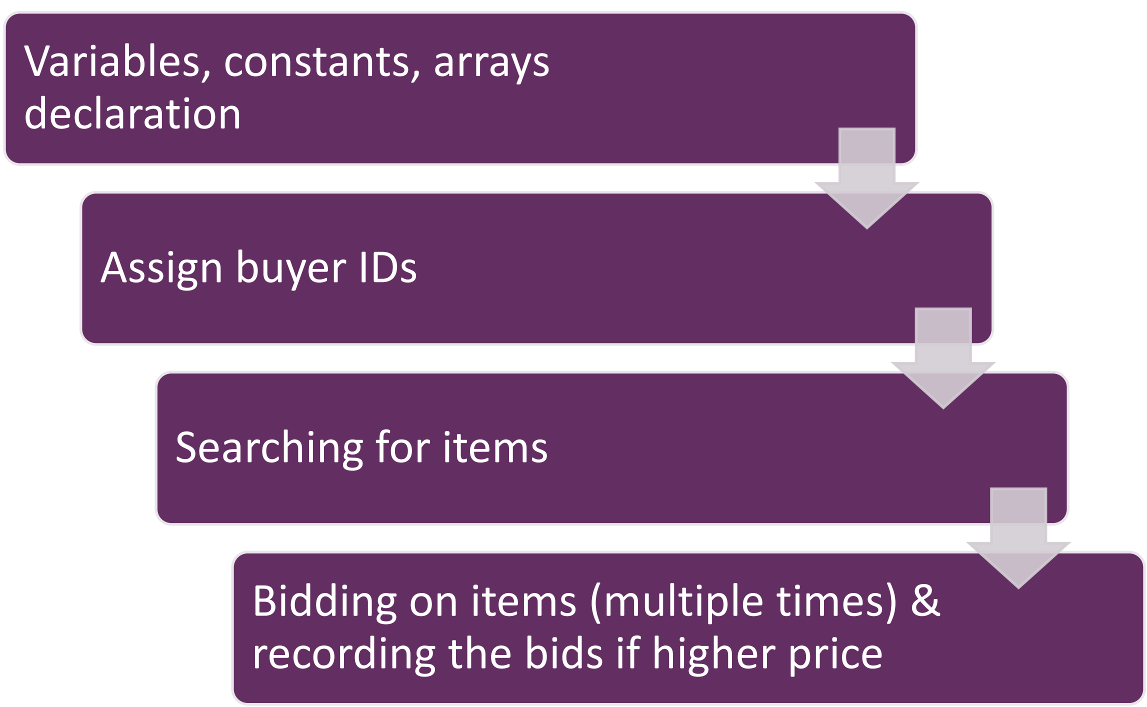
All 3 tasks are a part of one big problem i.e. setting up an auction system with bidding and results calculation. The tasks are built incrementally and each task uses the code/algorithm of the previous task. Hence the general flow and explanation of each task is provided separately in the following diagram.

Task 1



Task 2

** Task 1 identifiers will be used directly in this task without re-declaration*



Task 3

** Task 1 & 2 identifiers will be used directly without re-declaration*

- Variables, constants, arrays declaration
- Identification of sold items, calculation and display of 10% amount to be paid
- Identification and display of items with price which didn't reach the reserve price
- Identification and display of items that recieved no bids
- Display the summary of items sold, not sold & received no bids

Features of this Pre-Release

- Use of arrays data structure
- Overall Complexity: Easy / **Intermediate** / Challenging

Approach to Solution

Like every algorithm, there can be many possible approaches to solve these tasks depending upon the understanding of person. We are listing down the key points that reflect our understanding and we'll solve these tasks according to following **assumptions**.

- In Task1, the total items will be set as **constant** since the auction should have at least 10 items.
- In Task1, the item number needs to be **unique**. We will discuss **two possible ways** to achieve this and will implement the easier method.
- In Task2, the buyer will be assigned a unique ID programmatically (see the explanation on next page).
- In Task2, the buyer can search for items by entering particular item number.

Color Codes

The pseudocode uses different colors to represent keywords for easier understanding. These color codes are listed below.

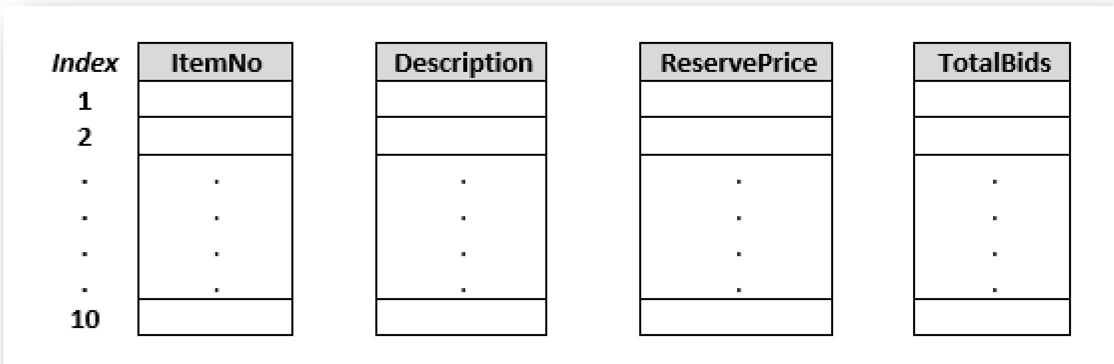
Begin / End	**BLACK**
Variable declaration and datatypes	**LIGHT BLUE**
Selection statements (IF and CASE)	**RED**
Input and Output	**GREEN**
Loop (REPEAT-UNTIL)	**PURPLE**
Strings/Text messages and variables	**BLACK**
Loop (FOR-NEXT)	**PINK**

Explanation of Algorithm of Tasks

The explanation of the algorithms used in each task is listed below.

Task 1

In this task we have to set up the auction by taking input related to items in the auction. As written clearly in the pre-release, there should be at least 10 items in the auction so all of our arrays will be of 10 elements at least. The information that we need to input are **item number, reserve price & its description** while **number of bids** will be set to zero for all items. All this information will be stored in 1D arrays using a FOR loop.



Now the challenging part in this task is to ensure that item number is unique. This can be done either **programmatically or manually**. If we choose the programmatic way, then we can utilize the power of **FOR loop counter variable** to generate unique numbers and assign them directly as item number. In this method user will not input the item number itself and the item numbers are guaranteed to be unique.

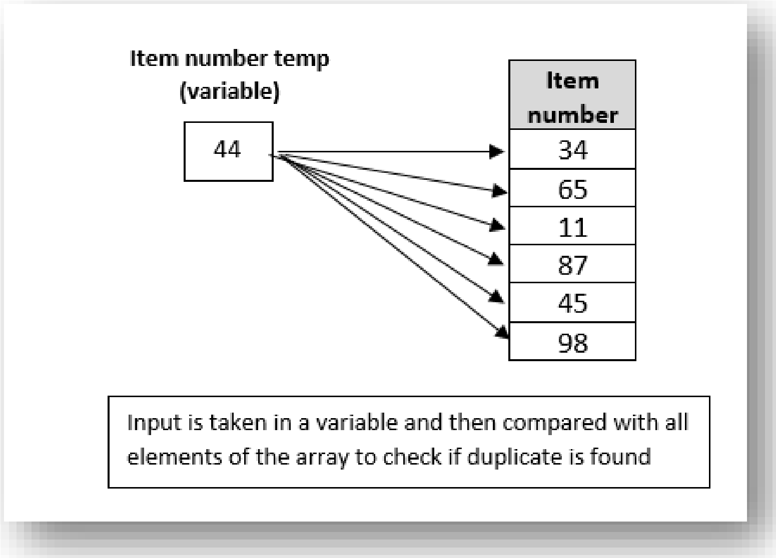
Example code

```
FOR count ← 1 TO 10
        Itemno[count] ← count
        .
        .
NEXT count
```

The loop counter variable **"count"** will generate a new number on each iteration which will be assigned as item number

 Prepared by: Blitz Computing

The alternative method is the **manual input method**. In this case we will take input the item number from the user and use a FOR loop to iterate over all elements to verify that it not already in the array. If the item number is already in the array, then we will display the error and let the user re-enter the code. This method requires a nested loop to work properly and accurately.



<u>*Example code*</u>

```
REPEAT
    INPUT "Enter item number", t_itemno
    duplicate ← FALSE
    FOR count ← 1 TO 10
        IF itemno[count] = t_itemno THEN duplicate ← TRUE
    NEXT count
    IF duplicate = TRUE THEN
        PRINT "Error. Duplicate found"
    ELSE
        itemno[cnt2] ← t_itemno
        cnt2 ← cnt2 + 1
    END IF
UNTIL cnt2 = 10
```

Both the above listed methods can be used in the pre-release however we will prefer the programmatic method due to its simplicity.

Task 2

In this task, we will initially set up the buyer IDs using an array and display them. Since the buyer IDs are only used while bidding, we will simply generate them using a FOR loop (just like we generate item numbers in task 1).
Then the user will be prompted to search for an item to bid. There can be two possible ways to search for the items; **one** is to display all item details in the auction and **the other** is to display only particular item number detail. Any method can be used, however we will be using the latter method in which we will ask the user to input the item number and then use this **item number as an index number** of the arrays to display its description and current highest bid.
The user will then type his buyer ID and bid for the item; the bid will be checked to see that it should be greater than the last bid. If the user bid is successful, then the total number of bids for that item is incremented by 1 and the user will be prompted whether he want to continue bidding or not.

Task 3

In this task we have to find the items which have the bidding price greater or equal to their reserve price and then mark them as “sold”. This can be done easily by declaring an array of **boolean datatype** and then loop through all item numbers to check if they are entitled to be marked as sold.
Other information that needs to be calculated and displayed are as follows:

- Grand total of 10% amount of all items which are sold
- Item number and bid price of all items which remained unsold (i.e. their bidding price is less than reserve price)
- Item number of items which received no bids i.e. total bids = 0
- A summary of count of all items which are sold, unsold and receive no bid.

Task 1 Solution (Pseudocode)

** The total items in the auction is assumed to be 10 and all arrays are according to 10 elements.*

```
BEGIN
    CONST maxitems ← 10 AS INTEGER
    DECLARE itemno[1:10] , totalbids [1:10], count ← 0 AS INETGER
    DECLARE description [1:10] AS STRING
    DECLARE reserveprice [1:10] AS FLOAT

    FOR count ← 1 TO maxitems
        itemno[count] ← count
        totalbids[count] ← 0
        description[count] ← " "
        reserveprice[count] ← 0.0

        PRINT "Enter details for item number", count
        INPUT "Enter description", description[count]
        IF description[count] = " " THEN
            PRINT "Error. Description cannot be left blank"
        END IF
        INPUT "Enter reserve price", reserveprice[count]
        IF reserveprice[count] < 0 THEN
            PRINT "Error. Price cannot be in negative"
        END IF
    NEXT count

END
```

Efficiency of Algorithm

- Use of **CONSTANT** to hold fixed value of items
- Use of ARRAY to store item number, description, total bids and reserve price
- Initialization of all arrays with default values.
- Use of **IF** statement to validate input and print appropriate error messages when validation fails.

Pseudocode Explanation of Task 1

```
BEGIN
    CONST maxitems ←10 AS INTEGER
    DECLARE itemno[1:10] , totalbids ←[1:10], count ←0 AS INETGER
    DECLARE description ←[1:10] AS STRING
    DECLARE reserveprice ←[1:10] AS FLOAT

    FOR count ←1 TO maxitems
            itemno[count] ← count
            totalbids[count] ← 0
            description[count] ← " "
            reserveprice[count] ← 0.0

            PRINT "Enter details for item number", count
            INPUT "Enter description", description[count]
            IF description[count] = " " THEN
                    PRINT "Error. Description cannot be left blank"
            END IF
            INPUT "Enter reserve price", reserveprice[count]
            IF reserveprice[count] < 0 THEN
                    PRINT "Error. Price cannot be in negative"
            END IF
    NEXT count

END
```

Variables, constants & arrays declaration.

Loop to item details

Initialization of arrays with default values

Validation using IF statement

Task 2 Solution (Pseudocode)

** Identifiers from Task 1 has been underlined for easier readability*
*** Assuming that there are 15 buyers*

```
BEGIN
    DECLARE highestbid[1:10], buyerbid AS FLOAT
    DECLARE buyerno[1:15], itemsearch, t_buyerno AS INTEGER
    CONST maxbuyers ← 15 AS INTEGER
    DECALRE choice AS CHAR

    FOR count ← 1 TO maxbuyers
        buyerno[count] ← count
        PRINT "Buyer ID is", buyerno[count]
    NEXT count

    REPEAT
            INPUT "Enter the item number to show its detail", itemsearch
            IF itemsearch < 1  OR  itemsearch > maxitems THEN
                PRINT "Wrong item number"
            ELSE
                PRINT "Item Number", itemno[itemsearch]
                PRINT "Description", description[itemsearch]
                PRINT "Highest bid", highestbid[itemsearch]
            END IF

            INPUT "Enter your buyer number", t_buyerno
            IF t_buyerno < 1  OR  t_buyerno > maxbuyers THEN
                PRINT "Wrong buyer number"
            END IF

            INPUT "Enter your bid value", buyerbid
            IF buyerbid < highestbid[itemsearch] THEN
                PRINT "Your bid value must be higher than the last bid"
            ELSE
                highestbid[itemsearch] ← buyerbid
                totalbids[itemsearch] ← totalbids[itemsearch] + 1
                PRINT "You bid is accepted & recorded"
            END IF
            INPUT "Do you want to bid for another item? (y/n)", choice
    UNTIL choice = 'n'
END
```

Efficiency of Algorithm

- Use of **IF** statement for validation of user input and print appropriate error message when validation fails
- Use of REPEAT loop to ask user about re-bidding

Pseudocode Explanation of Task 2

```
BEGIN
    DECLARE highestbid[1:10], buyerbid AS FLOAT
    DECLARE buyerno[1:15], itemsearch, t_buyerno AS INTEGER
    CONST maxbuyers ← 15 AS INTEGER
    DECALRE choice AS CHAR
```

Variables and constants declaration.

```
    FOR count ← 1 TO maxbuyers
            buyerno[count] ← count
            PRINT "Buyer ID is", buyerno[count]
    NEXT count
```

Generate and display 15 buyer IDs

```
    REPEAT
            INPUT "Enter the item number to show its detail", itemsearch
            IF itemsearch < 1  OR  itemsearch > maxitems THEN
                    PRINT "Wrong item number"
            ELSE
                    PRINT "Item Number", itemno[itemsearch]
                    PRINT "Description", description[itemsearch]
                    PRINT "Highest bid", highestbid[itemsearch]
            END IF
```

Ask user for item number to show its detail. Wrong item number will be rejected and correct will show the details

```
            INPUT "Enter your buyer number", t_buyerno
            IF t_buyerno < 1  OR  t_buyerno > maxbuyers THEN
                    PRINT "Wrong buyer number"
            END IF
```

Validation of buyer ID

```
            INPUT "Enter your bid value", buyerbid
            IF buyerbid < highestbid[itemsearch] THEN
                    PRINT "Your bid value must be higher than the last bid"
            ELSE
                    highestbid[itemsearch] ← buyerbid
                    totalbids[itemsearch] ← totalbids[itemsearch] + 1
                    PRINT "You bid is sucessful"
            END IF
```

Bid will only be accepted if it is higher than the last bid.

```
            INPUT "Do you want to bid for another item? (y/n)", choice
    UNTIL choice = 'n'
END
```

REPEAT loop will make sure user is able to re-bid as many times as he want.

Task 3 Solution (Pseudocode)

** Identifiers from Task 1 & 2 has been underlined for easy readability*

```
BEGIN
    DECLARE sold[1:10] AS BOOLEAN
    DECLARE auctionfee ← 0.0 AS FLOAT
    DECLARE totalsold ← 0, totalunsold ← 0, nobids ← 0 AS INTEGER

    FOR count ← 1 TO maxitems
        IF highestbid[count] > reserveprice[count]  THEN
            sold[count] ← true
            totalsold ← totalsold + 1
            auctionfee ← auctionfee + (highestbid[count] * 0.1)
        ELSE IF totalbids[count] > 0  THEN
            sold[count] ← false
            totalunsold ← totalunsold + 1
            PRINT "Item number", itemno[count], "didn't reach reserve price"
            PRINT "Final bid", highestbid[count]
        ELSE
            sold[count] ← false
            nobids ← nobids + 1
            PRINT "Item number", itemno[count], "received no bids"
        END IF
    NEXT count

    PRINT "Total auction company fee is:", auctionfee
    PRINT "Total sold items are:", totalsold
    PRINT "Total unsold items are:", totalunsold
    PRINT "Total items without any bids are:", nobids

END
```

Efficiency of Algorithm

- Nested **IF** statement is used to identify sold, unsold and items with no bids
- Use of Boolean "sold" array to store the status of items in the auction

Pseudocode Explanation of Task 3

```
BEGIN
    DECLARE sold[1:10] AS BOOLEAN
    DECLARE auctionfee ←0.0 AS FLOAT
    DECLARE totalsold ←0, totalunsold ←0, nobids ←0 AS INTEGER
    FOR count ←1 TO maxitems
        IF highestbid[count] > reserveprice[count]  THEN
                sold[count] ←true
                totalsold ←totalsold + 1
                auctionfee ←auctionfee + (highestbid[count] * 0.1)
        ELSE IF totalbids[count] > 0  THEN
                sold[count] ←false
                totalunsold ←totalunsold + 1
                PRINT "Item number", itemno[count], "didn't reach reserve price"
                PRINT "Final bid", highestbid[count]
        ELSE
                sold[count] ←false
                nobids ←nobids + 1
                PRINT "Item number", itemno[count], "received no bids"
        END IF
    NEXT count

    PRINT "Total auction company fee is:", auctionfee
    PRINT "Total sold items are:", totalsold
    PRINT "Total unsold items are:", totalunsold
    PRINT "Total items without any bids are:", nobids

END
```

Variables and array declaration.

Loop to iterate on all items

Nested IF statement to find sold, unsold and un-bid items. Note that the second "ELSE IF" only checks for "totalbids" array because if the first "IF" statement is false then it is certain that the item receive bid less than reserve price.

Complete Pseudocode of all Tasks

```
BEGIN
    CONST maxitems ←10, maxbuyers ←15 AS INTEGER
    DECLARE itemno[1:10] , totalbids [1:10], count ←0, totalsold ←0, totalunsold ←0,
    nobids ←0  AS INETGER
    DECLARE description [1:10] AS STRING
    DECLARE reserveprice [1:10], highestbid[1:10], buyerbid, auctionfee ←0.0 AS FLOAT
    DECLARE buyerno[1:15], itemsearch, t_buyerno AS INTEGER
    DECALRE choice AS CHAR
    DECLARE sold[1:10] AS BOOLEAN

    FOR count ←1 TO maxitems
        itemno[count] ← count
        totalbids[count] ← 0
        description[count] ← “ “
        reserveprice[count] ← 0.0

        PRINT “Enter details for item number”, count
        INPUT “Enter description”, description[count]
        IF description[count] = “ “ THEN
            PRINT “Error. Description cannot be left blank”
        END IF
        INPUT “Enter reserve price”, reserveprice[count]
        IF reserveprice[count] < 0 THEN
            PRINT “Error. Price cannot be in negative”
        END IF
    NEXT count

    FOR count ←1 TO maxbuyers
        buyerno[count] ←count
        PRINT “Buyer ID is”, buyerno[count]
    NEXT count

    REPEAT
        INPUT “Enter the item number to show its detail”, itemsearch
        IF itemsearch < 1  OR  itemsearch > maxitems THEN
            PRINT “Wrong item number”
        ELSE
            PRINT “Item Number”, itemno[itemsearch]
```

```
            PRINT "Description", description[itemsearch]
            PRINT "Highest bid", highestbid[itemsearch]
        END IF

        INPUT "Enter your buyer number", t_buyerno
        IF t_buyerno < 1  OR  t_buyerno > maxbuyers THEN
            PRINT "Wrong buyer number"
        END IF
        INPUT "Enter your bid value", buyerbid
        IF buyerbid < highestbid[itemsearch] THEN
            PRINT "Your bid value must be higher than the last bid"
        ELSE
            highestbid[itemsearch] ← buyerbid
            totalbids[itemsearch] ← totalbids[itemsearch] + 1
            PRINT "You bid is sucessful"
        END IF
        INPUT "Do you want to bid for another item? (y/n)", choice
    UNTIL choice = 'n'

    FOR count ← 1 TO maxitems
        IF highestbid[count] > reserveprice[count]  THEN
            sold[count] ← true
            totalsold ← totalsold + 1
            auctionfee ← auctionfee + (highestbid[count] * 0.1)
        ELSE IF totalbids[count] > 0  THEN
            sold[count] ← false
            totalunsold ← totalunsold + 1
            PRINT "Item number", itemno[count], "didn't reach reserve price"
            PRINT "Final bid", highestbid[count]
        ELSE
            sold[count] ← false
            nobids ← nobids + 1
            PRINT "Item number", itemno[count], "received no bids"
        END IF
    NEXT count

    PRINT "Total auction company fee is:", auctionfee
    PRINT "Total sold items are:", totalsold
    PRINT "Total unsold items are:", totalunsold
    PRINT "Total items without any bids are:", nobids
END
```

Practice Questions

1. When you performed the tasks, you used variables. Write suitable declarations for two of these. State what you used each one for.
2. When you performed the tasks, you may have used arrays. Write suitable declarations for any two of these. State what you used each one for.
3. Write an algorithm to complete Task 1, using either pseudocode, programming statements or a flowchart.
4. Write an algorithm to complete Task 2, using either pseudocode, programming statements or a flowchart.
5. Write an algorithm to complete Task 3, using either pseudocode, programming statements or a flowchart. You should assume that Task 1 & Task2 has been already completed.
6. Explain how you performed unique validation check for item numbers in Task 1. You can include pseudocode or programming statements as part of your explanation.
7. Explain how your program identify items which receive no bids. You can include pseudocode or programming statements as part of your explanation.
8. Explain what changes would be required in your pseudocode of Task2 & Task3 to allow the auction house to identify the buyer who won the item. You can include pseudocode or programming statements as part of your explanation.
9. Comment on the efficiency of your design for Task 2.
10. Comment on the efficiency of your design for Task 3.

Comments & Feedback

This document is prepared by **Blitz Computing** to help students prepare for Computer Science 2210 Paper 2 as well as for teachers delivering the same to their students.

We would like to hear your **comments, feedback and suggestions** which will **motivate** us providing quality content to students and teachers. If you find any mistake, feel free to inform us!

Visit our Facebook page at: www.facebook.com/blitzcomputing
OR
Email us at: info@blitz.education

Thank You!

Blitz Computing Team.