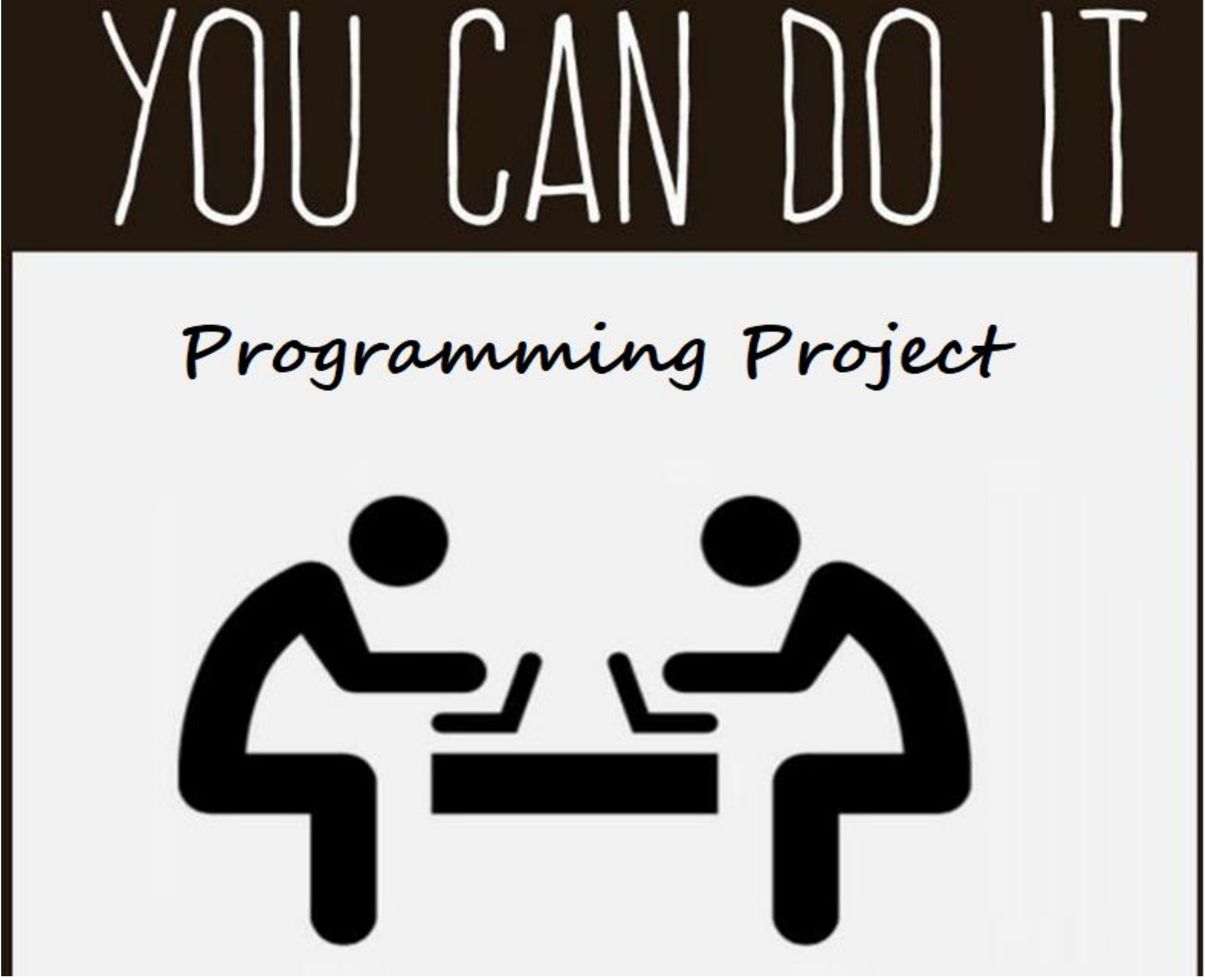


CS Programming Project Model Questions on Pre-release 2019

Name of participant: ………

Class/Section: ………

- Based on Pre-Release Material issued by CAIE for the Summer 2019 exams (This will also be used in Mock Exams)
- 100% related to syllabus
- It contains multiple tasks to design the solution, to code and to test their solutions.

<u>Here is a copy of pre-release material</u>

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.

Q1) When you performed the tasks, you may have used constants. [2+2]
Write suitable declarations for **two** of these. State what you used each one for.

Constant in task 1: MinItem=10

Use: To store fixed value of minimum number of items required to setup auction, i.e. 10

Constant in task 3: CompanyFeeRate=10

Use: To store rate of company fee applied on the items sold.

Q2) Fill in the following identifier table for task 1: [2+2]

Variable name	Data Type	Purpose
NoOfItems	INTEGER	Input and store number of items to be auctioned. At least 10 are required.
Count	INTEGER	To be used as loop counter and array subscript/index
ID	STRING	Temporary variable, To input Item ID. It may contain numerical ID like "101", leading zero like "001" and alphanumerical ID like "M01". After validation this ID is strored in ItemID Array

Q3) Arrays are data structures. State how do you decide size of array in task 1? [2]

At first number of items are input and validated, and then this number of items is used as array size, e.g. ItemID[1:NoOfItems]

Q 4) State arrays you have used in task 1 (3 arrays only): [3+3]

Data structure name	Data Type	Purpose
ItemID[1:NoOfItems]	STRING	To assign unique item number to each item using array. It may contain numerical ID like "101", leading zero like "001" and alphanumerical ID like "M01". Size of array is equal to number of items
Description[1:NoOfItems]	STRING	To input and store description of each item using array. Size of array is equal to number of items
ReservePrice[1:NoOfItems]	REAL	To input and store reserve price of each item using array. Size of array is equal to number of items
NoOfBids[1:NoOfItems]	INTEGER	To store number of bids of each item using array. Size of array is equal to number of items. Value of NoOfBid is incremented by 1 with each bid of the item.

Q5) Fill in the following identifier table for task 2 (2 variables only): [4]

Variable name	Data Type	Purpose
NoOfBuyer	INTEGER	Input and store number of buyers to bid.
BuyerIDToBid	INTEGER	Input and store buyer number who wants to bid
OfferedBid	REAL	Input and store buyer's offered bid
ToContinueAuction	STRING	To input from to decide that auction is to be continued or to be stopped

Q 6) State arrays you have used in task 2 (2 arrays only): [4]

Data structure name	Data Type	Purpose
ExistingBid[1:NoOfItems]	REAL	To input and store current highest bid id in 1D array
BuyerID[1: NoOfBuyer]	INTEGER	To assign unique ID to each buyer, starts from 1 and increment by 1 for each next buyer, maintain uniquness of buyer number

Q7) Fill in the following identifier table for task 3 (3 variables only): [3+3]

Variable name	Data Type	Purpose
CountSold	INTEGER	To count items whose existing bid has reached reserve price and have marked "SOLD"
CountNotSold	INTEGER	To count number of items those have received bids but haven't reached reserve price
Count0Bid	INTEGER	To count number of items those haven't received any bid
CompanyFee	REAL	To calculate 10% company fee of items who have sold
TotalFee	REAL	To calculate total company fee of all sold items

Q 8) Write an algorithm to complete **Task 1**, using **either** pseudo code, programming statements **or** a flowchart. Do not include declaration of variable. [6]

```
CONSTANT MinItems ← 10
 PRINT "Number of Items available for sale (atleast 10)"
 INPUT NoOfItems

  //Validation number of items
 WHILE NoOfItems < MinItems DO
          PRINT "Error:Auction could not be set up"
          PRINT "Enter No Of Items atleast 10"
          INPUT NoOfItems
    END WHILE

  //Setting up Arrays
    ItemID[1:NoOfItems]
    ReservePrice[1:NoOfItems]
    Description[1:NoOfItems]
    NoOfBids[1:NoOfItems]

 //Enterind detals of items on auction
    Index← 1
    REPEAT
          PRINT “Enter Item ID”
          INPUT ID
          SerachIndex←1
          IsFound ← False
          WHILE IsFound=False AND SearchIndex<=NoOfItem DO
                IF ID= ItemID[SearchIndex] THEN
                      IsFound=True
                ELSE
                      SearchIndex ← SearchIndex+ 1
                ENDIF
          ENDWHILE
          IF IsFound=False THEN
                ItemID[Index] ← ID
                INPUT “Enter description of Item “ Description[Index]
                INPUT “Enter reserve price of Item “ ReservePrice[Index]
                NoOfBid[Index] ← 0
                Index ← Index + 1
          ELSE
                PRINT “Error: Item ID is already taken, try another.”
          ENDIF
    UNTIL Index > NoOfItems
```

Q 9) Write an algorithm to complete **Task 2**, using **either** pseudo code, programming statements **or** a flowchart. You can assume that the task 2 is already completed.[6]

```
DECLARE NoOfBuyer, BuyerIDToBid:Integer
 DECLARE ToContinueAuction : String
 DECLARE OfferedBid : REAL
 ExistingBid[1: NoOfItems]
INPUT "Enter number of buyers for auction " NoOfBuyer
 BuyerID[1:NoOfBuyer]
 PRINT "Buyers setup"
 For Index ← 1 To NoOfBuyer
     BuyerID[Index] ← Index
 Next
 'Initialising current highest bid (ExistingBid[])with 0
 For Index ← 1 To NoOfItems
     ExistingBid[Index] ← 0
 Next Index
 'Starting Auction
 PRINT "Its auction time "
 REPEAT
     INPUT "Enter Buyer number " BuyerIDToBid
     While BuyerIDToBid > NoOfBuyer
         PRINT "Error: Buyer not found re-enter a valid buyer number "
         INPUT BuyerIDToBid
     End While
     Input "Enter Item number to bid " ID
      IsFound ← False
     SearchIndex ← 1
     While IsFound = False And SearchIndex <= NoOfItems
         If ItemID[SearchIndex] = ID Then
             ISFound = True
         Else
             SearchIndex = SearchIndex + 1
         End If
     End While
     If IsFound = True Then
         Print "Item description     : " , Description[SearchIndex]
         Print "Current highest bid  : " , ExistingBid[SearchIndex]
         INPUT "Enter offered bid : " OfferedBid
         If OfferedBid <= ExistingBid[SearchIndex] Then
             PRINT "Error: Your bid is lesser or equal to existing bid"
         Else
             ExistingBid[SearchIndex] ← OfferedBid
             NoOfBids[SearchIndex] ← NoOfBids[SearchIndex] + 1
         End If
     Else
         PRINT "Item not found try again"
     End If
     PRINT "Enter No to end auction, or press ENTER to continue "
     INPUT ToContinueAuction
 Until ToContinueAuction = "No"
```

Q 10) Write an algorithm to complete **Task 3**, using **either** pseudo code, programming statements **or** a flowchart. Do not include declaration of variable. You can assume that the task 1 & 2 are already completed. [6]

```
DECLARE CountSold, CountNotSold, Count0Bid : INTEGER
DECLARE MarkItem[1:NoOfItems] : STRING
DECLARE CompanyFee, TotalFee : REAL
CountSold ← 0
CountNotSold ← 0
Count0Bid ← 0
TotalFee ← 0
For Count ← 1 To NoOfItems
   If ExistingBid[Count] >= ReservePrice[Count] Then
      MarkItem[Count] ← "SOLD"
      CountSold ← CountSold + 1
      CompanyFee ← ExistingBid[Count] * 10 / 100
      TotalFee ← TotalFee + CompanyFee
   ElseIf ExistingBid[Count]<ReservePrice[Count] And NoOfBids[Count]>0 Then
      CountNotSold ← CountNotSold + 1
   Else
      Count0Bid ← Count0Bid + 1
   End If
Next
PRINT "Total company fee = " , TotalFee
PRINT "List of items have bids but not sold"
FOR Count ← 1 TO NoOfItems
   IF ExistingBid[Count] < ReservePrice(Count) AND NoOfBids[Count] > 0 THEN
      PRINT ItemNo[Count] , " , " , ExistingBid[Count]
   End If
Next
PRINT "List of items have no bid"
FOR Count ← 1 TO NoOfItems
   IF NoOfBids[Count] = 0 THEN
      PRINT ItemNo[Count]
   ENDIF
NEXT
    PRINT "Total number of items sold                  = " , CountSold
PRINT "Total number of items have bids but not sold = " , CountNotSold
PRINT "Total number of items have no bid            = " , Count0Bid
```

Q 11) Explain how do you validate that there are at least 10 items for auction. Include programming statement to support your explanation. [5]

A constant MinItems=10 is declared and then NoOfItems are input. Limit check is used to validate NoOfItems with WHILE loop.

Programming Statements:

```
CONSTANT MinItems ← 10
PRINT "Number of Items available for sale (atleast 10)"
INPUT NoOfItems
//Validation number of items
WHILE NoOfItems < MinItems DO
        PRINT "Error:Auction could not be set up"
        PRINT "Enter No Of Items atleast 10"
        INPUT NoOfItems
END WHILE
```

Q 12) Give **three** different data sets that could be used to check your validation rules for **Task 1**. Explain why you chose each data set. [2+2+2]

Data set 1:	30, 40, 20
Reason for choice:	This is normal data, it should be accepted by algorithm
Data set 2:	8, 6, 2
Reason for choice:	This is abnormal data, it should be rejected by algorithm
Data set 3:	10
Reason for choice:	This is extreme data, it should be accepted by algorithm

Q 13) Explain how do you ensure that item numbers are unique. Include programming statement to support your explanation. [4]

Explanation: Item ID is looked up in the array. If the entered ID is found in the array it is rejected and asked to enter another ID

Programming Statements:

```
PRINT “Enter Item ID”
INPUT ID
SerachIndex←1
IsFound ← False
WHILE IsFound=False AND SearchIndex<=NoOfItem DO
    IF ID= ItemID[SearchIndex] THEN
        IsFound=True
    ELSE
        SearchIndex ← SearchIndex+ 1
    ENDIF
ENDWHILE
IF IsFound=False THEN
    ItemID[Index] ← ID
    Index ← Index + 1
ELSE
    PRINT “Error: Item ID is already taken, try another.”
ENDIF
```

Q 15) Describe how do you assign unique number to each buyer in task 2 with the help of programming statements. [4]

Explanation: At first NoOfBuyer are input and then using count-controlled loop each value of loop counter is assigned to buyerID array

Programming Statements:

```
INPUT "Enter number of buyers for auction " NoOfBuyer
BuyerID[1:NoOfBuyer]
PRINT "Buyers setup"
For Index ← 1 To NoOfBuyer
        BuyerID[Index] ← Index
Next
```

Q 16) Explain how do you confirm that offered bid is greater than existing bid with the help of programming statement in task 2. [1 + 2+3]

Validation Rule: Limit check.

Explanation: OfferedBid is input from buyer and comapred with current highest bid (ExistingBid) of item if offered bid is greater than ExistingBid it will be selected and stored as new ExistingBid otherwise rejected.

Programming Statement for validation:

```
INPUT "Enter offered bid : " OfferedBid
If OfferedBid <= ExistingBid[SearchIndex] Then
   PRINT "Error: Your bid is lesser or equal to existing bid"
Else
   ExistingBid[SearchIndex] ← OfferedBid
   NoOfBids[SearchIndex] ← NoOfBids[SearchIndex] + 1
End If
```

Q 17) Give **two** different data sets that could be used to check validation rules in Q 16. Explain why you chose each data set. [2+2]

Data set 1: ExistingBid=150 OfferedBid=200

Reason for choice: It is a normal data it should be accepted by algorithm as offered bid is greater than existing bid

Data set 2: ExistingBid=200 OfferedBid=200

Reason for choice: It is a abnormal data it should be rejected by algorithm as offered bid is not greater than existing bid

Q 18) Comment on efficiency of code you have written in Q 16 above. [2]

It is an efficient code. It will compare offered bid with current highest bid (ExistingBid), whenever a buyer bids and item. It will accept only those bids which are greater than current highest bid (ExistingBid)

Q 19) Write down programming statements to input item number to bid in task 2 including validation check. [3]

```
Input "Enter Item number to bid " ID
'Validation of Item ID
IsFound ← False
SearchIndex ← 1
While IsFound = False And SearchIndex <= NoOfItems
   If ItemID[SearchIndex] = ID Then
      ISFound = True
   Else
      SearchIndex = SearchIndex + 1
   End If
End While
If IsFound = True Then
   Print "Item description     : " , Description[SearchIndex]
   Print "Current highest bid  : " , ExistingBid[SearchIndex]
   INPUT "Enter offered bid : " OfferedBid
Else
   PRINT “Item not found . Try again”
ENDIF
```

Q 20) Write down pseudo code to initialise COUNTing and TOTALing variables of task-3. [4]

```
CountSold ← 0
CountNotSold ← 0
Count0Bid ← 0
TotalFee ← 0
```

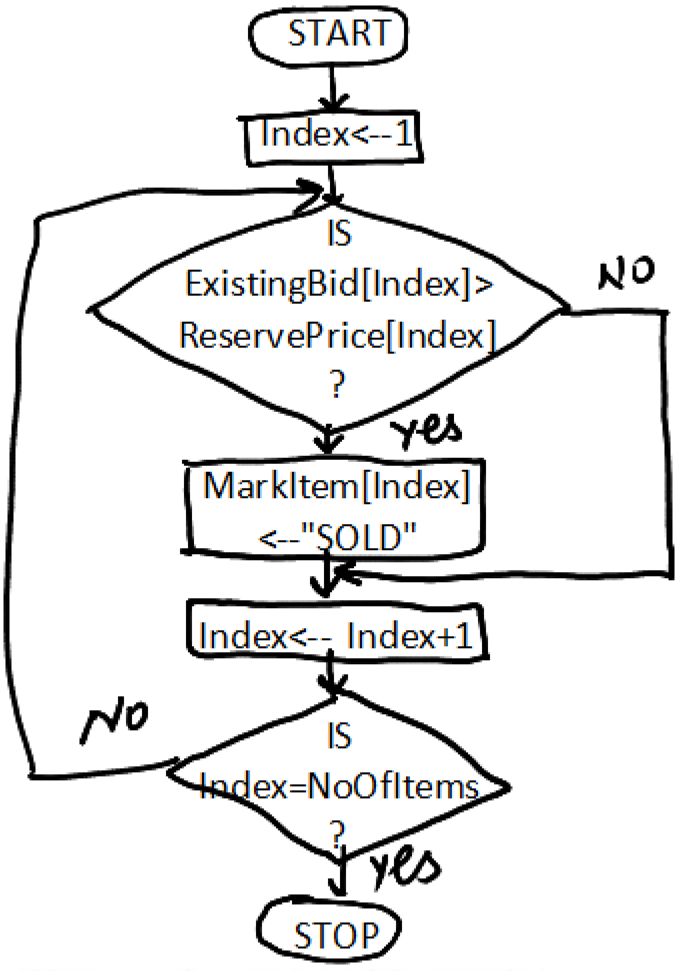
Q 21) Explain how do you mark an item “SOLD”. You should include programming statements to support your explanation. [5]

Explanation: At the end of auction ExistingBid of each item is compared with the ReservePrice of the item. If ExistingBid is greater or equals to ReservePrice item is marks as “SOLD”.

Programming Statements:

```
For Count ← 1 To NoOfItems
   If ExistingBid[Count] >= ReservePrice[Count] Then
      MarkItem[Count] ← "SOLD"
   End If
Next
```

Q 22) Draw program flowchart for the programming statements you have written in Q 21. [5]



Q 23) Explain how do you display item number of the item that has received the highest bid. You should include programming statements to support your explanation. [5]

Explanation: HighestBid is initialised with 0 and then ExistingBid of each item is compared with the HighestBid. And then ItemID of all items are displayed whose ExistingBid matches the HighestBid

Programming Statements:

```
HighestBid ← 0
FOR Index=1 TO NoOfItems
    IF ExistingBid[Index]>HighestBid THEN HighestBid← ExistingBid[Index]
NEXT Index
PRINT “List of items received highest bid in the auction”
FOR Index=1 TO NoOfItems
    IF ExistingBid[Index]=HighestBid THEN
        PRINT ItemID[Index]
    ENDIF
NEXT Index
```

Q 24) Explain how do you display item number of the item that has received the highest number of bids. You should include programming statements to support your explanation. [5]

Explanation: HighestNoOfBids is initialised with 0 and then NoOfBids of each item is compared with the HighestNoOfBids. And then ItemID of all items are displayed whose NoOfBids matches the HighestNoOfBids

Programming Statements:

```
HighestNoOfBids ← 0
FOR Index=1 TO NoOfItems
    IF NoOfBids[Index]>HighestNoOfBids THEN HighestNoOfBids← NoOfBids[Index]
NEXT Index
PRINT “List of items received highest number of bids in the auction”
FOR Index=1 TO NoOfItems
    IF NoOfBids[Index]=HighestNoOfBids THEN
        PRINT ItemID[Index]
    ENDIF
NEXT Index
```

Q 26) It is decided that item number is entered by seller. Explain how you ensure that the item numbers entered are unique. You should include programming statements to support your explanation. [5]

Explanation: Item ID is looked up in the array. If the entered ID is found in the array it is rejected and asked to enter another ID

Programming Statements:

```
PRINT “Enter Item ID”
INPUT ID
SerachIndex←1
IsFound ← False
WHILE IsFound=False AND SearchIndex<=NoOfItem DO
    IF ID= ItemID[SearchIndex] THEN
        IsFound=True
    ELSE
        SearchIndex ← SearchIndex+ 1
    ENDIF
ENDWHILE
IF IsFound=False THEN
    ItemID[Index] ← ID
    Index ← Index + 1
ELSE
    PRINT “Error: Item ID is already taken, try another.”
ENDIF
```

Q 25) Explain how do you search and confirm that item number entered by buyer is valid or invalid in task 2. You should include programming statements to support your explanation. [5]

Explanation: ItemID to bid is entered by buyer and then looked up in array of ItemID, If found then accepted otherwise rejected

Programming Statements:

```
Input "Enter Item number to bid " ID
'Validation of Item ID
IsFound ← False
SearchIndex ← 1
While IsFound = False And SearchIndex <= NoOfItems
   If ItemID[SearchIndex] = ID Then
      ISFound = True
   Else
      SearchIndex = SearchIndex + 1
   End If
End While
If IsFound = True Then
   Print "Item description     : " , Description[SearchIndex]
   Print "Current highest bid  : " , ExistingBid[SearchIndex]
   INPUT "Enter offered bid : " OfferedBid
Else
   PRINT “Item not found . Try again”
ENDIF
```