**Design and Analysis of Data Structures and Algorithms 2018 – 19**

**Tasks to be completed**

1. Identify and justify the choice of data structures and algorithms to be used to implement the four storage warehouses (data storage) and manipulate the data – not more than 500 words.

**2.** **Design the software and provide an efficient algorithm that would support the above requirements. Write relevant and detailed pseudo code.**

3. Implement the system with the following functionality.

a. Add an item to one of the warehouses

b. Initially reject an item if no available capacity exists in a single warehouse

c. Identify items to be moved to different warehouses to make space for the new item

d. Move the selected items to alternative warehouses

e. Reject a new item if its value is higher than the total insured value of a single warehouse

f. Reject a new item if the remaining storage capacity (in insured value) across all four locations is less than the value of the item

4. Produce well-structured and fully commented code. 5. Complete appropriate test data sets to thoroughly test your software and use them to test your system thoroughly.

**Task 2:WRITE REVELANT AND DETAILED PSEUDOCODE**.

**MAIN MENU**

Read data from file or csv into Warehouses

Initialise menu item selected =0

DO

Display Main Menu

Get user menu item selected

Get valid integer

IF menu item selected ==1

**Add an item to Warehouse**

ELSE IF menu item selected ==2

**Display Warehouse items**

ELSE IF menu item selected ==3

**Move Warehouse items**

ELSE IF menu item selected ==4

**Remove Warehouse items**

ELSE IF menu item selected ==5

**Restore Discarded items**

ELSE IF menu item selected ==6

**Search an Item**

ELSE IF menu item selected ==7

End program

END IF

WHILE menu item selected !=7

**Function: ADD AN ITEM TO WAREHOUSE**

Initialise selected warehouse =0

Initialise repeat add item ==’Y’

WHILE repeats add item ==’Y’

Clear console screen

FOR ALL warehouses

Display name

Display remaining insurance

Display number of elements

Display Warehouses selection

Get user selected warehouse

**Get valid integer**

**Add records**

User Selected warehouse

Get user input for repeat add item

**Get valid yes or no**

IF repeat == ‘Y’ THEN

**Add an item to warehouse**

ELSE

Return to main menu

END IF

END WHILE

**Function: ADD RECORDS**

Initialise item not discarded = True

Display Warehouse

Display warehouse name

Display remaining insurance

Display items

Get inputted fields

Request item number

**Check id existence(all items, add records)**

**Get valid integer**

Request item price

**Get correct price**(arguments: warehouse, item price, items not rejected)

IF item not rejected is True THEN

Request item description

**Get valid item description**

Add item to selected Warehouse

Decrease selected warehouse and overall insurance

Print success message

Update screen

END IF

**Function: DISPLAY WAREHOUSE ITEMS**

Initialise choice=0

Initialise repeat =’Y’

WHILE repeat ==’Y’

Clear screen

FOR ALL warehouses

Display name

Display remaining insurance

Display number of elements

END FOR

Display warehouses and get user selection

**Get valid integer**

Print selected warehouse info

IF number of items ==0 then

Display warehouse empty message

Return

ELSE

Display warehouse name

Display warehouse remaining insurance

Display total elements

END IF

WHILE last item not displayed

Display item number

Display item description

Display item price

END WHILE

Get user input for repeat

**Get valid yes or no**

IF repeat == ‘Y’ THEN

**Display warehouse items**

ELSE

Return to main menu

END IF

END WHILE

**Function: MOVE WAREHOUSE ITEMS**

Initialise restore items = False , initialise move items = True, initialise repeat = ‘Y’

WHILE repeat ==’Y’

FOR ALL warehouses

Display name

Display remaining insurance

Display number of elements

END FOR

Get user selection

**Get valid integer**

Print selected warehouse info

IF number of items of selected warehouse ==0 then

Display warehouse empty message

Return

ELSE

Display warehouse name

Display warehouse remaining insurance

Display total elements

END IF

**Pick items to be moved**(selected warehouse, restore items, move items)

Get user input for repeat

**Get valid yes or no**

IF repeat == ‘Y’ THEN

**Move warehouse items**

ELSE

Return to main menu

END IF

END WHILE

**Function: PICK ITEMS TO BE MOVED(selected warehouse, restore items, move items)**

Initialise select more items = ‘Y’, User item selection =0 Initialise picked items =empty

IF restore items is False THEN

While select more items is ==’Y’ and warehouse not empty

Sort warehouse items by price in ascending order using **Insertion sort**

Display warehouse items and already selected items

Display warehouse info

Sort warehouse by item number in ascending order using **Merge-sort**

Get user item selection

Get valid item number

**Get valid integer**

Add selected item to picked items

IF restore items is False then

Increase warehouse insurance by selected item price

END IF

Display changes and get user select more items input

IF select more items == ‘Y’ THEN

**Pick items to be moved**

ELSE

Get user destination selection

**Get valid integer**

**Select destination**(arguments: destination warehouse, selected warehouse, picked items, restore items)

END IF

ELSE

Add selected items to discarded items warehouse

Delete picked items list

END IF

**Function: SELECT DESTINATION(arguments: destination warehouse, selected warehouse, picked items, restore items)**

Calculate picked items worth

Initialise items not rejected =True

IF destination warehouse not valid then

**Return items to selected warehouse**

ELSE

Check destination warehouse remaining insurance

**Get correct price**(arguments: destination warehouse, selected items worth, items not rejected)

IF items not rejected is True then

WHILE picked items list not empty

Add picked item to destination warehouse

Decrease destination and overall warehouse insurance by item worth

END WHILE

ELSE

**Return items to selected warehouse**(selected warehouse, picked items, selected items worth, restore items)

END IF

END IF

**Function: RETURN ITEMS TO SELECTED WAREHOUSE(arguments: selected warehouse, picked items, selected items worth, restore items)**

IF restore items is False then

Add picked items to selected warehouse

Decrease selected warehouse and overall insurance

ELSE

Add picked items to discarded items warehouse

END IF

**Function: REMOVE WAREHOUSE ITEMS**

Initialise restore items = False

Initialise move items = False

Initialise repeat = ‘Y’

WHILE repeats == ‘Y’ then

FOR ALL warehouses

Display name

Display remaining insurance

Display number of elements

END FOR

Get user selection

**Get valid integer**

Print selected warehouse info

Display warehouse name

Display warehouse remaining insurance

Display total elements

WHILE last item not displayed

Display item number

Display item description

Display item price

END WHILE

**Pick items to be moved**(selected warehouse, restore items, move items)

Get user input for repeat

**Get valid yes or no**

IF repeat == ‘Y’ THEN

**Remove warehouse items**

ELSE

Return to main menu

END IF

**Function: RESTORE WAREHOUSE ITEMS**

Initialise repeat ==’Y’

Initialise restore items = True

Initialise move items = True

IF number of discarded warehouse elements ==0

Print(“There are no items to be restored)

ELSE

WHILE repeat ==’Y’ and number of discarded items > 0

**Pick items to be moved**(selected warehouse, restore items, move items)

Update screen

**Get valid yes or no**

IF repeat == ‘Y’ THEN

**Restore warehouse items**

ELSE

Return to main menu

END IF

END WHILE

END IF

**Function: SEARCH ITEM**

Initialise search attempts =3

Initialise repeat =’Y’

WHILE search attempts >0 and repeat ==’Y’

Display remaining attempts

Request item id

**Get valid integer**

**Interpolation search item** id across all warehouses and return index

IF item found then

Print Warehouse position

Print Item number

Print item description

Print Item Price

ELSE

Print item not found

Reduce attempts by 1

END IF

Get user input for repeat

**Get valid yes or no**

IF repeat == ‘Y’ THEN

**Search Items**

ELSE

Return to main menu

END

END WHILE

**Function: GET VALID YES OR NO**

Initialise flag = True

WHILE flag is True

Request yes or no input

IF Input == ‘Y or N’

Return input

ELSE

Display error message(“Wrong input”)

END WHILE

**Function: GET VALID INTEGER(minimum, maximum)**

Initialise flag = True

WHILE flag is True

Request integer input

IF input not integer then

Display value error message

ELSE IF input < minimum and > maximum

Display out of range message

ELSE

Return input

END IF

END WHILE

**Function: CHECK ID EXISTENCE()**

WHILE item exist already

Get correct id

**Get valid integer**

Search item across all warehouses

**Binary search(all items, start, end, id, item non-existent)**

IF item exist already

Display item exists message

ELSE

Set item exist already = False

Return id

END IF

END WHILE

**Function: BINARY SEARCH (all items, starting index, ending index, id, item non-existent)**

IF ending index >= starting index then

Middle point index = (ending index +s starting index)/2

IF value at middle point of all items == id

Return id

ELSE

IF value at middle point index of all items > id

Recursively call binary search with

Starting index = 0

Ending index = middle index value-1

ELSE

Recursively call binary search with

Starting index = middle+1

Ending index =ending index

END IF

END IF

ELSE

Display item not found message

Set item non-existent to False

END IF

**Function: INTERPOLATION SEARCH (warehouse list, id)**

Get length of warehouse list

Initialise start =0

Initialise end = length of warehouse list -1

IF warehouse list empty then

EXIT

WHILE item not found

Calculate position of item using the formula

position = int (start+(float((end-start) /(warehouse[end].item\_number- warehouse[start].item\_number))\*(id- warehouse[start].item\_number)))

IF item found at index position

Return position

IF id > item at index position

Set start index to position +1

ELSE

Set end to position -1

END IF

END IF

IF id not found

Return False

END IF

**Function: GET CORRECT PRICE (arguments: warehouse, item price, items not rejected)**

Initialise flag =True

WHILE Flag == True

Validate item price

IF item price < 1 or > 2000000000

Display item price out of range message

Set flag to False

Set items not rejected to False

ELSE IF item price > Warehouse remaining insurance

Reject item

Ask if user wants to apply algorithm

**Get valid yes or no**

IF input == ‘Y’ THEN

**Algorithm (warehouse, item price)**

IF algorithm is FALSE then:

Display item rejected message

Set flag to False

Set items not rejected to False

END IF

ELSE

Display item discarded message

Set flag to False

Set items not rejected to False

END IF

END IF

END WHILE

Return item price

**Function: GET VALID ITEM DESCRIPTION()**

Initialise Flag = True

WHILE flag ==True

Request item description input

IF input < 1 or > 45

Display item description length error

ELSE

Return item description

END IF

END WHILE

**Function: ALGORITHM (Selected Warehouse, item price)**

Display suggested moveable items to User

Get user input yes or no

**Get valid yes or no**

IF input == ‘Y’:

Copy all warehouses objects into one list

Sort warehouses in ascending order by remaining insurance

Remove selected warehouse from list

**Move items to increase insurance** (selected warehouse, allwarehouses objects, item price)

ELSE

Return False

END IF

**Function: MOVE ITEMS TO INCREASE INSURANCE (selected warehouse, allwarehouses objects, item price)**

Initialise flag = False

Initialise redistribute items = False

WHILE item price > selected warehouse insurance and redistribute is False

FOR ALL warehouses in allwarehouses objects

IF warehouse in allwarehouses objects insurance > selected item price

Remove item from selected warehouse

Add item to warehouse with enough insurance

Increase selected warehouse insurance by item price

Decrease warehouse in allwarehouses objects insurance by item price

Set flag = True

Recursively call **move items to increase insurance()**

END IF

END FOR

IF Flag == False then

Sort allwarehouses objects in descending order by insurance

Backup allwarehouses objects warehouses items

Delete allwarehouses objects warehouses items

Reset warehouses in allwarehouses objects remaining insurance

WHILE selected warehouse insurance < item price

Sort selected warehouse by item price in ascending order

Add selected warehouse first item to backup

Increase selected warehouse insurance by first item price

Remove first item from selected warehouse

END WHILE

Initialise redistribute items to True

END IF

IF redistribute items is True

Sort backup items in descending order

FOR ALL warehouses in allwarehouses objects

IF warehouse in allwarehouses objects insurance > selected item price

Add item to warehouse in allwarehouses objects

Decrease warehouse insurance

ELSE

Add item to discarded items warehouse

END IF

END FOR

END WHILE

**Function: INSERTION SORT(Selected warehouse)**

Initialise j=0

Initialise i =0

WHILE i < length of selected warehouse

Insert element at position i of selected warehouse to temp position 0

Set j = i-1

WHILE item price of temp > selected warehouse item price at position j and j>0

Insert item at position j at j+1

Decrease j by 1

END WHILE

Insert temp item into selected warehouse at position j+1

Increase i by 1

END WHILE

**Function: MERGE SORT (Selected warehouse)**

IF length of selected warehouse <=1

Return selected warehouse

ELSE

Calculate middle point of selected warehouse

Middle = selected warehouse length /2

Sort the first half of selected warehouse

Sort the second half of selected warehouse

Merge the two halves back

Set first half index =0

Set second index =0

WHILE first half index < length of first half and

second index < length of second half

Compare item number at first half index to item number at second half index

IF smaller

Add item to result list

Increment first half index by 1

ELSE

Add item to result list

Increment second half index by 1

END IF

END WHILE

Copy remaining items to result list

Return result list

END IF