CIS 2101 Machine Problem #2
BST and Priority Queue
April 29, 2023

Machine Problem RUBRICS					
Note: At minimum, the program should run. No compilation errors.					
Criteria	**Percentage**	**Scale**			
		3	2	1	0
Meets program specifications	70%	All of the function modules are implemented correctly. (All 3 Problems are answer correctly)	No. of Problems answered correctly : 2	No. of Problems answered correctly : 1	No. of Problems answered correctly : 0
Readability	15%	Code is organized and easy to follow and 100% of the agreed coding conventions are followed	Code is fairly easy to read and 80% of the agreed coding conventions are followed	Code is readable only by somehow who knows what the code does and 60% of the agreed coding conventions are followed.	Code is poorly organized and less than 60% of the agreed coding conventions are followed
Efficiency	15%	Code is efficient without sacrificing readability. No unnecessary variables are used and no unnecessary and redundant statements. Code is at its optimum.	Code is 80% efficient without sacrificing readability. At most 20% of the code can be improved in terms of running time, storage, and lines of code	Code is 60% efficient and somehow unnecessarily long. 40% of the code can be improved in terms of running time and storage, and lines of codes.	Code is done in brute force manner.

Problem Description:

The program implements the following ADTs: BST and Priority Queue.

A BST of chocolate records is organized in internal memory using cursor-based implemented. Each element is uniquely identified through the ID number. The virtual heap can be shared by 2 or more BST's.

An unsorted set of chocoloate records, organized in internal memory using an array implementation, is converted into a minheap using heapify technique. The technique starts with thelowest level parent going up to the root node, and making each subtree a Partially Ordered Tree (POT).

INSTRUCTIONS: Complete the given partial program.

1) **Create a folder in the drive D or drive E named: CIS2101_Prefinal**
2) **Create a .c program with filename : StudCode_Lastnamexx.c //xx is the first 2 letters of the firstname**

Example: A12_DelaCruzju.c

Write a program that will implement and CALL all the functions whose prototypes are given.

The function prototypes and specification are given. The program will be checked based on the correctness of each of the subproblems. NOTE: Partial/incomplete program is given.

The following functions prototypes have to be implemented.

Function Prototypes	Description
Problem # 1:	
BST createMagicalBSTvheap(VHeap *VH);	**Complete Code is provided. The function creates and populates a BST implemented using cursor-based representation in memory. In effect, the function organizes the Virtual heap. The RC field in the node of the virtual heap is used to link the nodes, and a -1 means the node is the last node in the list of nodes.**
void displayVHeap(VHeap V);	**Partial Code is provided. The function displays the indexes of the nodes in the virtual heap, together with the product ID, LC and RC fields. Included in the display is the value of the available cell.** **Given below is the partial expected output.**

void displayProduct(product P);	**Complete Code is provided. The function displays the information about the a given product.**
void displayBSTAscending(BST B, VHeap V);	**The function displays the chocolate records in the BST is ascending order according to product ID. This function calls displayProduct().**

Expected Output for Problem #1

```
Problem #1::
------------

Details of the Virtual Heap ::
------------------------------
Available Index  :: 6

     Index    Prod ID       LC Field       RC Field
     -----    -------       --------       --------
       0       1356            8              2
       1       1703           -1             -1
       2       1550            3             13
       3       1450            4             10
       4       1310           -1             -1
       5       1688           -1             -1
       6       1901           -1             11
       7       1701            0              9
       8       1109           -1             14
```

. . .

//Partial output

```
Elements of BST B
1109    Patchi    50    99.75     35
1201    Kitkat    50    97.75     40
1284    Lindt     100   250.75    15
1356    Ferrero   200   250.75    85
1310    Nestle    100   124.50    70
1450    Ferrero   100   150.50    50
1455    Tango     75    49.50     100
1550    Cadbury   120   200.00    30
1601    Meiji     75    75.50     60
1688    Guylian   50    99.75     35
```

. . .

//Partial output

Problem # 2: Implements an insert operation in a BST represented in memory using cursor-based representation.

void insert(BST *B, VHeap *VH, product P);	**The function inserts a new chocolate record in the BST if there is still space in the virtual heap which is shared by other BST's and if the given record does not yet exist in the BST.** **Possible messages in this function:** **"Unsuccessful Insert of Product (value of prod ID): Virtual Heap is Full."** **"Unsuccessful Insert of Product (value of prod ID): Element exists."** **"Successful Insert of Product (value of prod ID)."**

Expected Partial Output for Problem #2

```
Problem #2::
------------
Unsuccessful Insert of Product 1109: Element exists.
Successful Insert of Product 1356.
```

. . .

```
Elements of BST new
1356    Ferrero    200    250.75    85
1807    Mars       100    150.75    20
```

Problem # 3: Heapifies a given list of elements.

minHeap populate();	**Complete Code is provided. The function creates and populates a minHeap.**
void displayHeap(minHeap HL);	**Partial Code is provided. The function displays all products in the heap. This function calls displayProduct().**
void swapProduct(product *x, product *y);	**Complete Code is provided. The function swaps the value of two product product record. This is used in heapfiSubtree().**
void heapifySubtree(minHeap *H, int subroot);	**This function will heapify the subtree by pushing the root node of the subtree to its proper position in the minheap. This function is called by heapify().**
void heapify(minHeap *H);	**This function heapifies the entire collection by calling heapifySubtree() starting from the lowest level parent down to the root node.**

Expected Partial Output for Problem #3

```
Problem #3::
------------

List/Heap :: 12 Elements
========================
1701    Toblerone 50    90.75     80
1356    Ferrero   200   250.75    85
1807    Mars      100   150.75    20
1109    Patchi    50    99.75     35
1550    Cadbury   120   200.00    30
1201    Kitkat    50    97.75     40
1450    Ferrero   100   150.50    50
1601    Meiji     75    75.50     60
1284    Lindt     100   250.75    15
1310    Nestle    100   124.50    70
1455    Tango     75    49.50     100
1688    Guylian   50    99.75     35
```

//Before heapify

```
List/Heap :: 12 Elements
========================
1109    Patchi    50    99.75     35
1284    Lindt     100   250.75    15
1201    Kitkat    50    97.75     40
1356    Ferrero   200   250.75    85
1310    Nestle    100   124.50    70
1688    Guylian   50    99.75     35
1450    Ferrero   100   150.50    50
1601    Meiji     75    75.50     60
1701    Toblerone 50    90.75     80
1550    Cadbury   120   200.00    30
1455    Tango     75    49.50     100
1807    Mars      100   150.75    20
```

//After heapify