

CIS 2101 Machine Problem # 1

List, Set and Dictionary

March 25, 2023

Machine Problem RUBRICS					
<b>Note:</b> 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 5 Problems are answer correctly )	No. of Problems answered correctly : 3 to 4	No. of Problems answered correctly : 1 to 2	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: List, Set and Dictionary.

A set of chocolate records is represented in internal memory using cursor-based implemented. Each element is uniquely identified through the ID number. The set is SORTED is ascending order according to ID.

The set of chocolate records is converted into a dictionary of chocolate records which is represented in internal memory using OPEN HASHING. Each group in the dictionary is SORTED in ascending order according to ID and is represented in memory using cursor-based implementation. Note: The cursor set and the dictionary are sharing the same virtual heap, i.e. the program has only 1 virtual heap.

The Open Hash Dictionary is converted into a Closed Hash Dictionary. To avoid displacement, a 2-pass loading is implemented, i.e. synonyms are temporarily stored in an array implemented List. In the 2<sup>nd</sup> round or pass the synonyms stored in the List are added to the Close Hash Dictionary.

INSTRUCTIONS: Complete the given partial program.

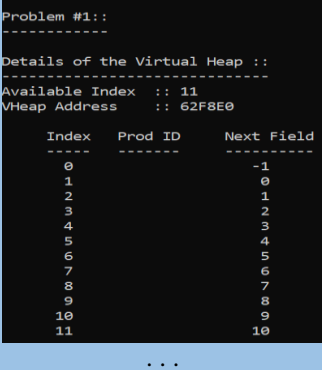
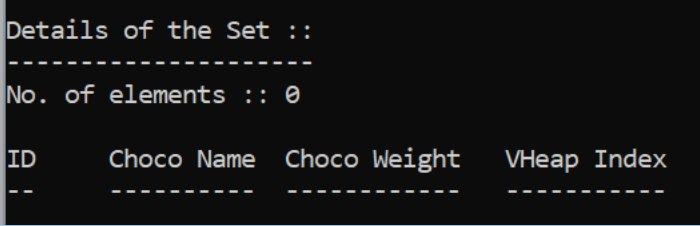
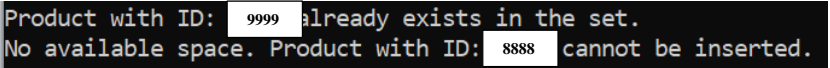
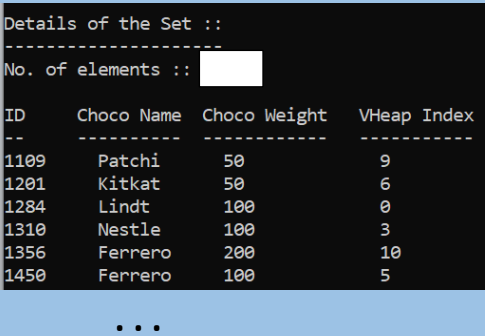
- 1) Create a folder in the drive D named: CIS2101\_StudCode
- 2) Create a .c program with filename : StudCode\_Lastnamexx.c //xx is the first 2 letters of the firstname

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: Part of the program is given BELOW.

The following functions prototypes have to be implemented.

Function Prototypes	Description
Problem # 1: Initializes and displays the virtual heap and the cursor set.	
void initVHeap(VHeap *VH);	The function initializes the virtual heap by linking the nodes and making the last index in the array as the value of avail variable. In addition and for printing purposes, initialize the ID number to be equal to 4 spaces.
cursorSet initCursorSet(VHeap *VH);	The function initializes and returns a set implemented using cursor-based. The set contains a pointer to an initialized or existing virtual heap.

<code>void displayVHeap(VHeap V);</code>	<b>Partial Code is provided.</b> The function displays the indexes and the next field values of the virtual heap. Included in the display are the values of the available cell and the address of the virtual heap in internal memory. Given below is the expected output. Note: Vheap Address may differ.
<code>void displaySet(cursorSet A);</code>  Note: Given below are sample output displays.	<b>Partial Code is provided.</b> The function displays the contents of the cursor set. Details of each choco record are displayed HORIZONTALLY. The details include the ID number, Choco Name, Choco Weight, and its INDEX in the virtual heap. Total number of elements is also displayed.
Expected Output for Problem #1	
	
Problem # 2: Populates the cursor set sorted in ascending order according to ID and displays its contents	
<code>int mallocInVHeap(VHeap *VH);</code>	The function removes the first available cell or node in the virtual heap and returns the index of the removed cell or node to the calling function.
<code>void insertSorted(cursorSet *A, product P);</code>	This function inserts a product record in its appropriate position in the sorted set if the product does not yet exist. The ID number uniquely identifies the elements (products) and is the basis for ascending order arrangement of the elements.  If the product already exists or there is no available space, it displays:  
<code>void populateSet(cursorSet *A);</code>	<b>Partial Code is provided.</b> The function populates the set with list elements provided in the function. This invokes function insertSorted() hence this cannot be completed if insertSorted() is not created.
Expected Partial Output for Problem #2	
	
Problem # 3: Converts the cursor set into an Open Hash dictionary and displays its contents. It also displays the empty set.	
<code>int openHash(char *IDen);</code>	This function returns the hash value of a given ID number by adding its NUMERIC digits and reducing its value appropriate to the size of the hash table. Example: ID numbers “1204” and “1343” have hash values of 7 and 1 respectively.
<code>openDic initOpenDict(VHeap *VH);</code>	This function initializes the dictionary to be empty. The dictionary contains a pointer to an existing virtual heap (which is also used by cursor set).
<code>openDic convertToOpenDict(cursorSet *A);</code>	This function removes/deletes the elements from the cursor set and inserts them in the dictionary. Each group in the dictionary is sorted in ascending order according to ID.
<code>void displayOpenDict(openDic D);</code>	<b>Partial Code is provided.</b> The function displays the contents of the dictionary. It displays the group number and the ID numbers of elements in each group. It also displays the total number of elements in the dictionary.

Expected Partial Output for Problem #3

```
Problem #3::
-----
Details of the Open Hash Dictionary::
-----
No. of elements :: 
GROUPS      ID Numbers
-----
Group[0] :: 1450
Group[1] :: 1109      1550      1703
Group[2] :: 
Group[3] :: 
Group[4] :: 1201
Group[5] :: 1284      1310      1356      1455

...

```

```
Details of the Set ::
-----
No. of elements :: 0

ID      Choco Name      Choco Weight      VHeap Index
--      -

```

Problem # 4: Displays the contents of the dictionary and the virtual heap after performing 3 delete operations.

<code>void freeInVHeap(VHeap *VH, int ndx);</code>	The function returns back to the heap the node whose index is given by the variable ndx. This is equivalent to the free() function in C.
<code>void deleteDic(Dictionary *D, char *IDen);</code>	This function deletes the element in the dictionary bearing the given ID number.

Expected Partial Output for Problem #4

```
Problem #4::
-----
Product with ID: 9999 is successfully deleted.
Product with ID: 8888 is not in the dictionary

```

```
Details of the Open Hash Dictionary::
-----
No. of elements :: 
GROUPS      ID Numbers
-----
Group[0] :: 1450
Group[1] :: 1109      1550
Group[2] :: 
Group[3] :: 
Group[4] :: 1201
Group[5] :: 1284      1310      1356      1455

...

```

```
Details of the Virtual Heap ::
-----
Available Index  :: 4
VHeap Address   :: 62F8E0

Index  Prod ID  Next Field
-----
0      1284      3
1      -1        -1
2      1455      -1
3      1310      10
4      -1        1
5      1450      -1
6      1201      -1
7      1807      -1

...

```

Problem # 5: Creates a closed hashing dictionary implemented using open hashing dictionary and displays the contents of the closed hash table.

<code>int closeHash(char *ID);</code>	This function returns the hash value of a given ID number by adding its NUMERIC digits and reducing its value appropriate to the size of the hash table. Example: ID numbers “1284” and “1701” have hash values of 3 and 9 respectively.
<code>void initCloseDict(closeDic CD);</code>	This function initializes the close Hash Dictionary to be empty using the ID field.
<code>closeDic * convertToCloseDict(openDic *D);</code>	The function will convert the open hash dictionary into a closed Hash dictionary using a 2 pass loading, i.e. synonyms are temporarily stored in the array implemented List. In the 2 <sup>nd</sup> round of insertions, elements stored in the List are added to the close hash dictionary.
<code>void displayCloseDict(closeDic CD);</code>	<b>Partial Code is provided.</b> The function displays the contents of the Closed Hash Table.

Expected Partial Output for Problem #5

```
Problem #5::
-----
Details of Closed Hash Dictionary ::
-----
Index  ChocoID  Choco Name
-----
0      1550      Cadbury
1
2
3      1284      Lindt
4      1201      Kitkat
5      1310      Nestle
6      1356      Ferrero
7      1455      Tango
8      1807      Mars
9      1701      Toblerone

...

```

```
Details of the Virtual Heap ::
-----
Available Index  :: 11
VHeap Address   :: 62F8E0

Index  Prod ID  Next Field
-----
0      -1        6
1      -1        -1
2      10       10
3      0        0
4      1        1
5      4        4
6      8        8
7      2        2
8      9        9

...

```