CS 2101 Machine Problem #2
List, Set, Dictionary and Graph – December 11,2023

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.

There are 3 problems which are independent of each other and stored in 3 different files.

Problem 1: **Populates a list and makes it a set.**

Problem 2: **Close dictionary with 2 Pass-loading.**

Problem 3: **Finds the edges of the minimum cost-spanning tree using Prim's Algorithm starting at the vertex which is inputted by the user.**

Names of files to be submitted:

Problem 1: **Final_Prob_01_lastnameXX.c // Your Lastname and XX [1st two letters of your first name]**

Problem 2: **Final_Prob_02_lastnameXX.c**

Problem 3: **Final_Prob_03_lastnameXX.c**

Programming Instructions:

1) **The return keyword will only be used if the function's return type is not void and there should 1 return statement only.**
2) **Function should not have a break or exit statement.**
3) **Proper indention should be followed and the code is readable.**
4) **Only efficient code gets full credit.**

The following functions prototypes have to be implemented.

Function Prototypes	Description
Problem # 1: Creates, Populates, and displays a list of chocolate records. Converts the list into a set and displays the set.	
`ProdList populateProdList();`	**Partial Code is provided. This function populates the newly created linked list and returns it to the calling function. The order of the chocolate records in the array is the same as the order of the chocolate records in the linked list.**
`void displayProdList(ProdList L);`	**Partial Code is provided. This function displays the ID, Chocolate name and weight of each chocolate record in the given list. In addition, it also displays the no. of elements displayed.**

<table>
<tr><td></td><td>Partial Output:

```
Details of the List ::
--------------------

ID     Choco Name  Choco Weight
--     ----------  ------------
1701   Toblerone   135
1356   Ferrero     200
1109   Patchi      50
1550   Cadbury     120
```

...</td></tr>
<tr><td><code>void makeListToSet(ProdList L);</code></td><td>This function will make the given list a set by removing duplicates, i.e. the first occurrence of a chocolate record is retained and any duplicate of the record is removed. Note that chocolate records are uniquely identified through its ID.</td></tr>
<tr><td colspan="2">Problem # 2: Populates a product set. Initializes the dictionary to be empty, and inserts the elements of the product set into the dictionary implementing a 2 Pass-Loading, i.e. all the synonyms are inserted only into the dictionary in the second pass to avoid displacement of elements. The function displayCloseDict() is called 3 times to display an empty dictionary, with elements but without synonyms and with all the elements.</td></tr>
<tr><td><code>ProdSet populateProdSet();</code></td><td>Complete Code. This function will return a populated ProdSet.</td></tr>
<tr><td><code>int closeHash(char *ID);</code></td><td>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.</td></tr>
<tr><td><code>void initCloseDict(closeDic CD);</code></td><td>This function initializes the close Hash Dictionary to be empty using the ID field.</td></tr>
<tr><td><code>void displayCloseDict(closeDic CD);</code></td><td>Partial Code is provided. The function displays the contents of the Closed Hash Table.
Given below are partial displays.
<table>
<tr><th>Empty Dictionary</th><th>Dictionary without synonyms</th></tr>
<tr><td>

```
Index ChocoID   Choco Name
----- -------   ----------
0     empty
1     empty
2     empty
3     empty
4     empty
5     empty
```

...</td><td>

```
Details of Closed Hash Dictionary ::
-----------------------------------

Index ChocoID   Choco Name
----- -------   ----------
0     1356      Ferrero
1     1807      Mars
2     empty
3     empty
4     1201      Kitkat
5     1310      Nestle
```

...</td></tr>
</table></td></tr>
<tr><td><code>closeDic * createCloseDict(ProdSet S);</code></td><td>Given a product set, the function will create closed Hash dictionary using a 2 pass loading, i.e. synonyms are temporarily stored in a different product set. In the 2nd round of insertions, elements stored in the temporary set are added to the close hash dictionary. If collision occurs, a linear probing/hashing is used.
In addition, this function will call displayCloseDict() 3 times:
1) The dictionary is empty
2) The dictionary with all the elements, without the synonyms
3) The dictionary with all the elements + synonyms</td></tr>
<tr><td colspan="2">Problem # 3: Finds the edges of the minimum cost-spanning tree using Prim's Algorithm starting at the vertex which is inputted by the user.</td></tr>
<tr><td><code>void populateGraph(graphType G);</code></td><td>Complete Code.</td></tr>
<tr><td><code>primMST primAlgo(graphType graph, int Vertex);</code></td><td>Given a graph and the start vertex Vertex, the function will return a list of edges in the MST.</td></tr>
<tr><td><code>void displayPrimMST(primMST tree);</code></td><td>Partial Code is given.

Sample output at starting vertex 3.

```
Start Vertex is 3
 Edge Cost
(3,4)     1
(3,2)     2
(4,5)     3
(3,0)     4
(0,1)     1
Minimum Cost : 11
```
</td></tr>
</table>