Practice problems for heap, hashing, and graph

1. Change the max heap into a min heap. In a min heap, each node's key is smaller than the keys in its child nodes. Insert key 1 to the following min heap H by modifying the insertion algorithm in our lecture note about heap. Show the contents of the array H whenever there is a change. H[]={2,3,7,4,9,8,15, 13, 10}.

   Specifically, you should first: 1) describe the updated insertion pseudocode for min heap. 2) apply the code to this input and show the contents of H whenever there is a change.

2. Consider a hash table of size 11 with all keys as integers. The hash function is h(key)=key % 11. Collisions are resolved by quadratic probing. At one stage, the hash table has the following content.

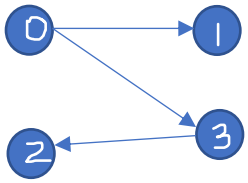
23
90
34
61
16
10

   (a) What is the number of probs to search for key 16?
   (b) Show the contents of the hash table after the key = 100 has been inserted.
   (c) What is the problem of using quadratic probing to handle collision?

3. Programming problem. Implement both the recursive and non-recursive version of DFS on a DAG (what is DFS? What is DAG? Review the lecture note). The input is a file containing the adjacency matrix. For example:

   4
   0 1 0 1
   0 0 0 0
   0 0 0 0
   0 0 1 0

   represents a graph of 4 nodes (the first line with one integer) and it looks like below:



In addition, your program should ask the user to input a starting node. I will provide several test graphs. The algorithms/pseudocodes can be found in the lecture note about the graph. The output is simply the traversal of the graph using DFS and thus I expect to see a list of nodes. For example, the output of calling DFS using 0 as the starting node in the "recursive DFS" should be:
0 1 3 2
If using non-recursive version of DFS, the output is: 0 3 2 1

Note that when a node has multiple adjacent nodes, we always process them by their numerical order.

In this document, I will provide some help about using the STL vector as a queue.

https://www.cplusplus.com/reference/vector/vector/

1. You can use vector<vector<int>> to define a 2D array (review our note about graph)
   Some examples:

```
vector<vector<int>> vect
    {
        {11, 12, 23},
        {44, 59, 60},
        {7, 800, 99}
    };


    for (int i = 0; i < vect.size(); i++)
    {
        for (int j = 0; j < vect[i].size(); j++)
        {
            cout << vect[i][j] << " ";
        }
        cout << endl;
    }
```

2. You can use vector::back() and vector::pop_back() to implement the "pop_back"f function in the pseudocode. The first function will save the last

element. The second function will remove the last element. An example can be found here:
https://www.cplusplus.com/reference/vector/vector/pop_back/