CSCI 2270 Midterm 2 Practice

Problem 1

Task:

Write a program that prints the roof of BST. Roof of the BST is the

```
   / \
  /   \
 /     \
/       \
```

created by the root and the left side of the tree and the right side of the tree

Requirements:

Implement a `printRoof` function to print the roof of a given binary tree

```
void BST::printRoof() // example declaration
```

Examples:

- For the following tree `printRoof()` should print `3 15 20 30 40`.

```
           20
         /     \
       15        30
      /  \        \
     3    18        40
      \
      7
     /
    6
   /
  5
 /
4
```

- For the following tree `printRoof()` should print `18 30 40`

```
            18
               \
                30
               /  \
              20   40
```

- For the following tree `printRoof()` should print `20 30`

```
      30
     /
    20
```

Write a main function that creates a BST of integers, then calls the function defined above to demonstrate the method. Call it on multiple trees to show that it works in all cases, and print out the results.

Problem 2

Task:

Write a program that given 2 arrays check if they are a subset of each other.

Requirements:

Implement a `isSubset` function that checks if all elements of arr2 are in arr1

```
bool isSubset(int arr1[], int arr2[]) // example declaration
```

You can assume that there are distinct elements in the array.

Hint: You can use hashtables to check if elements from a list exist in the other

Examples:

- Calling `isSubset` on `[1,2,3,4,5]` and `[2,3,1]` would return `true` as `2, 3 and 1` are in the first array

- Calling `isSubset` on `[1,2,3,4,5]` and `[2,3,1,6]` would return `false` as `2, 3 and 1` are in the first array but `6` is not

Problem 3

Task:

In graph theory, a bridge is an edge of a graph whose deletion disconnects the graph. In other words, if you delete a bridge, then you increase the number of connected components. Write a program that creates an *undirected* graph of char's and a function that determines whether a given edge is a bridge.

Requirements:

Implement an `isBridge` function that returns true if the edge between two vertices is a bridge and false otherwise.

```
bool Graph::isBridge(char x, char y); // example declaration
```

You may assume that there exists an edge between `x` and `y`.

Examples:

- Consider the following graph (Your graph will have char type rather than int). `isBridge(1,2)` and `isBridge(1,0)` should both return `false`, but `isBridge(2,3)` should return `true` since deleting the edge from 2 to 3 disconnects the graph.

Write a main function that creates an undirected graph of char’s, then calls the function defined above to demonstrate the method. Call it on multiple edges of the graph to show that it works in all cases, and print out the results.