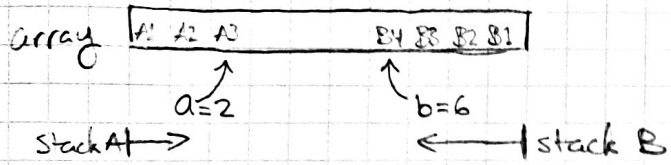
Data Structures 1 - HW4 - Abraham Murciano

1) Given an array of size n, the way to implement 2 stacks would be to have 1 stack starting at position 0, going upwards, using one variable to store the index of the 'top' element of that stack. Then the second stack would begin at position n-1 and elements would be pushed onto the previous elements of the array. Again there would be a variable to store the index of the 'top' element of that stack. (see illustration below)



2)
```
class stack {
    queue Q;
    int size;
    void push(int x){      // complexity of push is Θ(size)
        Q.enqueue(x);
        for(int i=0; i<size; i++){
            Q.enqueue(Q.dequeue());
        }
        size++;
    }
    int pop(){      // complexity of pop is Θ(1)
        if(isEmpty()){
            throw "Stack is empty; cannot pop";
        }
        size--;
        return Q.dequeue;
    }
    bool isEmpty(){      // complexity of isEmpty is Θ(1)
        return size == 0;
    }
};
```

3) a)
```
void foo(queue& Q1, queue Q2){ int Q1size = Q1.size();
    for(int i=0; i<Q1size; i++){
        int temp = Q1.dequeue();
        if(search(Q2, temp)){
            Q1.enqueue(temp);
        }
    }
}
bool search(queue Q, int x){
    for(int i=0; i<Q.size(); i++){
        if(x == Q.dequeue()){
            return true;
        }
    }
    return false;
}
```

b) complexity is $O(n \times m)$ because complexity of search is $O(m)$, and the loop in foo runs n times, each time calling seach, so $O(n \times m)$