```
//This class is used for developing a Stack.
class Stack {
    Node top;
    Node next;

//The push function creates a new node for the old top.
//It then creates a new node object and assigns a name.
//The next node to the top is the old top.
    void push(String previous) {

        Node oldtop = top;
        top = new Node();
        top.name = previous;
        top.next = oldtop;
    }

//The top function creates a new variable of type Node to the
old top.
//The top variable is equal to the next node and will return the old top.
    String pop() {
        Node oldTop = top;
        top = top.next;
        return oldTop.name;
    }
    //IsEempty checks if the top is null which means the stack is empty.
    public boolean isEmpty() {
        return top == null;
    }
}
//The node class creates a node which has the values name and next.
//name is associated with the value of the Node. Node next is the next variable
class Node {
    String name;
    Node next;
}

class Queue {
    Node head, tail;

    void enqueue(String inComingChar) {

        Node oldtail = tail;
        tail = new Node();
        tail.name = inComingChar;
```

```
        tail.next = null;

        if (isEmpty()) {
            head = tail;
        } else {
            oldtail.next = tail;
        }

    }

    String dequeue() {
        String fetVal = "";
        if (!isEmpty()) {
            fetVal = head.name;
            head = head.next;
            return fetVal;
        }
        if (isEmpty()) {
            return "No value";
        }
        return fetVal;

    }

    boolean isEmpty() {
        if (head == null) {
            return true;
        } else {
            return false;
        }
    }
}
public class Homework1{

    public static void main(String[] args){
        int countOfPalindrome = 0;
        int count = 0;
        String myArray[];
        myArray = new String[666];
        try{
        File magicItems = new File("magicitems.txt");
        Scanner myScanner = new Scanner(magicItems);

        //This while loops scans the next line and puts it into the next index o
        while (myScanner.hasNextLine()) {
            String data = myScanner.nextLine();
```

```
            myArray[count] = data;
            count++;
          }
          myScanner.close();
        } catch (FileNotFoundException e) {
          System.out.println("An error occurred.");
          e.printStackTrace();
        }


    //This is where all of the above code comes together
    //initalize a new Stack and Queue object
    Stack newStack = new Stack();
    Queue newQueue = new Queue();
    String queueValue;
    String stackValue;

    //Lines 121-129 will push and enqueue the charatacers of the strings onto a
    for (int i = 0; i < myArray.length;i++){
        for (int j = 0; j < myArray[i].length(); j++){
            //System.out.println(Character.toString(myArray[i].charAt(j)));
            if (Character.toString(myArray[i].charAt(j)).equals(" ")){
                continue;
            }
            newStack.push(Character.toString(myArray[i].charAt(j)).toLowerCase()
            newQueue.enqueue(Character.toString(myArray[i].charAt(j)).toLowerCas

        }
//Lines 132-148. The Queue and Stack will check if a Node type is still in them.
//If there is a Node in them they will dequeue and pop them out of the stack and
//If the head of the queue and stack is null, then it will create a new object o
        while(newQueue.head != null && newStack.top != null){
            queueValue = newQueue.dequeue();
            stackValue = newStack.pop();

            if (!queueValue.equals(stackValue)){
                newQueue = new Queue();
                newStack = new Stack();
                break;
            }
//print out the palindromes.
            if (queueValue.equals(stackValue) && newStack.isEmpty()){
                System.out.println(myArray[i]);
                countOfPalindrome++;
            }
```

```
            }
        }
        System.out.println("There are " + countOfPalindrome + " paliondromes");

        }
}
```