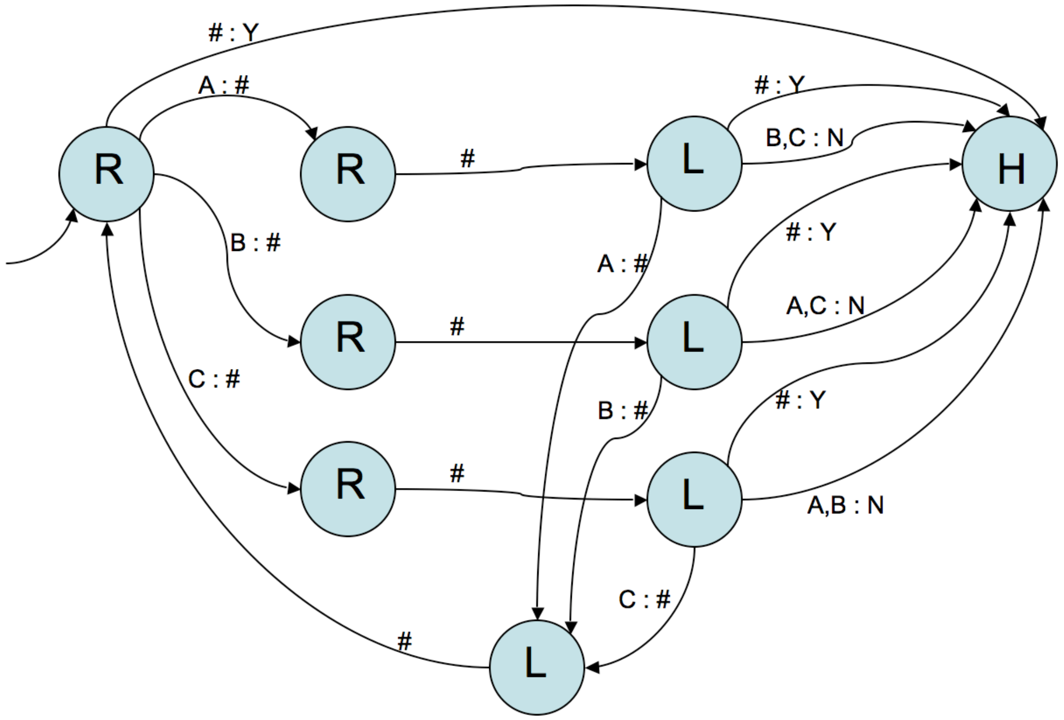
COS 126 Fall 2006 Exam 2 Solutions

1. Reading down the list: G, C, A, B, E, F

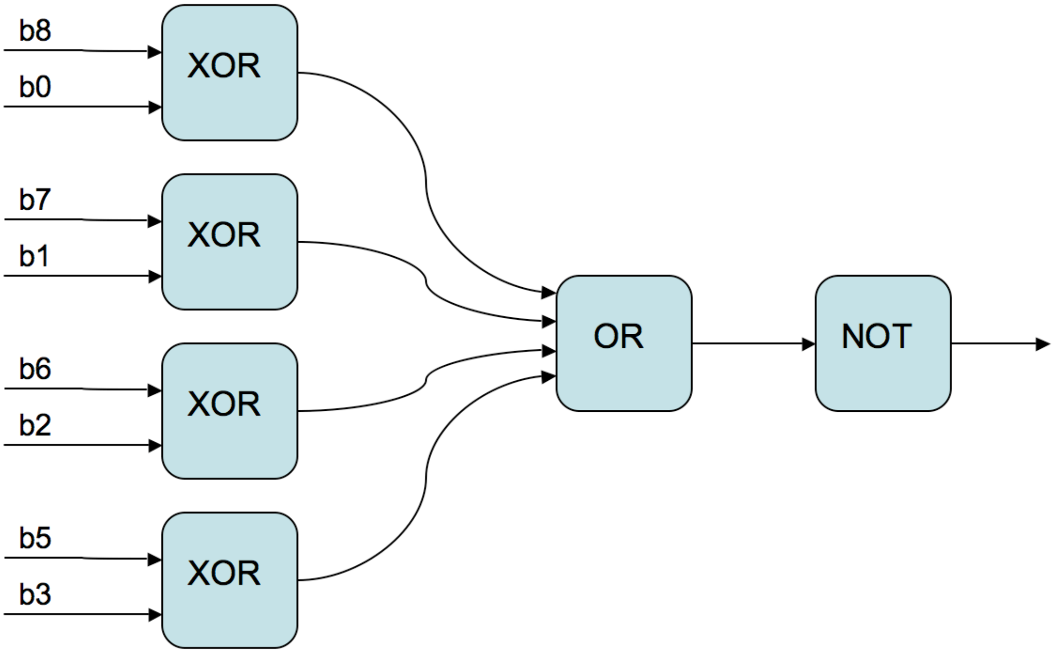
2.

```
public class Palindrome {

  public static boolean isPunc(char c){
    if ((c==' ')|| (c==',')||(c =='.')||(c==':')||
        (c==';')||(c=='!')||(c=='?')) return true;
    else return false;
  }

  public static void main(String[] args) {

       String s = args[0].toLowerCase();
       boolean pal = true;
       for (int i=0, j=s.length()-1; i < j; i++, j--) {
         while (isPunc(s.charAt(i))) i++;
         while (isPunc(s.charAt(j))) j--;
         if (s.charAt(i) != s.charAt(j)) pal = false;
                }
       System.out.println(pal);
   }
}
```

3.



4. Let the bits be named b8, b7, . . . , b0



5.

```
public void reverse()
{
    Node tmp;

    for (Node n = first; n != null; )
    {
        tmp = n.next;
        n.next = n.prev;
        n.prev = tmp;

        n = tmp;
    }

    tmp = first;
    first = last;
    last = tmp;
}
```

6.

```
public ArrayQueue(int max) {
     N = max;
     a = new String[N];
     back = 0;
     front = 0;
}

public Boolean isEmpty() {
     return (back == front);
}

public enqueue(String s) {
     a[back] = s;
     back = (back + 1) % N;
}

public String dequeue() {
     String s = a[front];
     front = (front + 1) % N;
     return s;
}
```

7. Reading down the list: T, T, F, F, T, F, T, F, T, T

8.

```
public class Client {
  public static void main (String[] args) {
    int N = Integer.parseInt(args[0]);
    Interval[] intervals = new Interval[N];

    for (int i = 0; i < N; i++) {
      intervals[i] = new Interval(StdIn.readDouble(),
                     StdIn.readDouble());
    }

    for (int i = 0; i < N; i++){
      for (int j = i + 1; j < N; j++) {
        if (intervals[i].intersects(intervals[j]))
               System.out.println(intervals[i] +
                " intersects " + intervals[j]);
      }
    }
  }
}
```

9. Had you taken COS 126 you would have learned that your idea for the Verifier is, sadly, impossible to realize, for it is not even possible to write a computer program that would say, correctly and always, whether an arbitrary other program would eventually halt on a given input. Your Verifier would have this impossible ability, and so cannot exist. Please don’t fire me.