1. Turing Machines
2. Circuits

   a) Truth Table

   | S | D0 | D1 |  
   |---|---|---|---|
   | 0 | 0 | 0 | 0 |
   | 0 | 0 | 1 | 0 |
   | 0 | 1 | 0 | 1 |
   | 0 | 1 | 1 | 1 |
   | 1 | 0 | 0 | 0 |
   | 1 | 0 | 1 | 1 |
   | 1 | 1 | 0 | 0 |
   | 1 | 1 | 1 | 1 |

   b)
public class AudioPlayer {
    private Node start; // first song
    private Node end; // last song
    private Node cur; // current song

    // a Node is a node in a linked list
    private class Node {
        private String filename;
        private Node next;
    }

    public AudioPlayer() {
        // create empty linked list
        start = null; end = null; cur = null;
    }

    // add song to the end of list
    public void addSong(String title) {
        Node tmp = new Node();
        tmp.filename = title;
        tmp.next = null;
        if (start == null)
            end = start = tmp;
        else {
            end.next = tmp;
            end = end.next;
        }
    }

    // play and print the current song
    private void playCur() {
        System.out.println(“Now Playing: “ + cur.filename);
        StdAudio.play(cur.filename);
    }

    // play and print all songs
    public void playAll() {
        cur = start;
        while (cur != null) {
            this.playCur();
            cur = cur.next;
        }
    }
}
// make requested song current, and play it and print it
// assume it is in the list
public void skipToThisSong(String filename) {
    cur = start;
    while (cur != null) {
        if (cur.filename.equals(filename)) {
            this.playCur();
            break;
        }
        cur = cur.next;
    }
}

// main (test client)
public static void main(String[] args) {
    AudioPlayer player = new AudioPlayer();
    // read song filenames from StdIn and store in the
    // linked list
    while (!StdIn.isEmpty()) {
        String name = StdIn.readString();
        player.addSong(name);
    }
}
}
4. **TOY**

```plaintext
//pop
60: 7101 R[1] <- 1 constant 1
66: EF00 goto R[F] TOY’s return statement
```

5. **Object Oriented Programming**

```plaintext
public class STNew {

    private STLite list[];

    public STNew() {
        list = new STLite[10];
        for (int i = 0; i < 10; i++) list[i] = new STLite();
    }

    public void put(int key, String value) {
        list[key % 10].put(key, value);
    }

    public String get(int key) {
        return list[key % 10].get(key);
    }
}
```
6. Queues

    public static Queue<Integer> QueueMerge(Queue<Integer> r, Queue<Integer> s) {
        Queue<Integer> q = new Queue<Integer>();
        while ( !r.isEmpty() || !s.isEmpty() ) {
            if (r.isEmpty()) q.enqueue(s.dequeue());
            else if (s.isEmpty()) q.enqueue(r.dequeue());
            else if (r.peek() < s.peek()) q.enqueue(r.dequeue());
            else q.enqueue(s.dequeue());
        }
        return q;
    }

7. Regular Expressions

a)  
   (i)  (A|B)(A|B) Answer: 4
   (ii) AB+ Answer: \infty
   (iii) AB* Answer: \infty
   (iv) AB Answer: 1

b) Answer: (i), (ii), (iii), and (iv)
Every regular expression specifies a set of strings that can be accepted by some deterministic finite state automaton.

c) In Java, the regular expression “\d” matches any digit. The equivalent is (0|1|2|3|4|5|6|7|8|9)

d) Each answer below is one or more of these strings.
   A: alphabet
   B: abracadabra
   C: babcock
   D: hubbub
   E: suburbia
   F: dabchick
   (i) ab Answer: ABCF
   (ii) abc Answer: CF
   (iii) a.*a Answer: AB
   (iv) .........* (eight dots) Answer: ABCEF
   (v) bu(b|r) Answer: DE
8. Theory

T The Church-Turing Thesis is called a thesis and not a theorem because it is a statement about the real world that cannot be formally proved.

T It’s possible to write a program that can decide whether another program solves the halting problem.

T In general, it is undecidable whether a Turing Machine will halt on a given input.

F In general, it is undecidable whether a Turing Machine will halt on a given input after at most n steps.

F If a problem is in P, then any program that solves that problem must run in polynomial time.

T If a problem is in P, then it’s possible to write a program that checks proposed solutions to that problem in polynomial time.

T If a problem is in NP, then it’s possible to write a program that checks proposed solutions to that problem in polynomial time.

T If P=NP, then the Traveling Salesperson Problem can be solved in polynomial time.

T If the Traveling Salesperson Problem can be solved in polynomial time, then P=NP.

T When a DFA is processing a particular input string, its running time will always be polynomial in the length of that input string.