1. Encapsulation, ADT (8 points)

```java
public class Account {
    private int balance;

    // constructor, initializing account balance to init
    public Account(int init) {
        this.balance = init;
    }

    // deposit amt into account
    public void deposit(int amt) {
        this.balance = this.balance + amt;
    }

    // withdraw amt from account if there is enough balance
    // otherwise, print an error message and withdraw nothing
    public void withdraw(int amt) {
        if (amt <= this.balance)
            this.balance = this.balance - amt;
        else
            System.out.println("Insufficient funds");
    }

    // transfer amt to the account b if there is enough balance
    // otherwise, print an error message and transfer nothing
    public void transfer(int amt, Account b) {
        if (amt <= this.balance) {
            this.balance = this.balance - amt;
            b.balance = b.balance + amt;
        }
        else
            System.out.println("Insufficient funds");
    }

    // get current balance
    public int getBalance() {
        return this.balance;
    }
}
```
2. Regular Expressions, Deterministic Finite State Automata (6 points)

a) The answer, iii) generates all desired strings and only desired strings.
   i) can generate a string that starts with b.
   ii) cannot generate a single a.
   iv) can generate a string that starts with b.
   v) cannot generate a single a.

b) The answer, i) accepts all desired strings and only desired strings.
   ii) accepts the empty string.
   iii) accepts strings that start with b.

3. Linked Lists (6 points)

(a) i) returns true
    ii) returns true
    iii) returns false
    iv) returns false

(b) **linky_dink** returns true for a null-terminated linked list. It returns false for a circular linked list, even if the circular part is preceded by a straight path.

(c) N

For a null terminated linked list, b will traverse each node once before the method returns true. For a circular linked list, b which is traveling twice as quickly as a, will catch up to a in a constant number of circuits of the length N list.

4. Analysis of Algorithms (3 points)

(a) \(N^2\)

The nested loop performs \(N\) times \(N\) additions.

(b) \(N\)

The loop performs \(N\) multiplications.

(c) \(\log N\)

The recursive method uses repeated squaring. It calls itself approximately \(\log N\) times, performing either one or two multiplications each call.

5. QuickSort (5 points)

(a) 

(b) \(N\log N\) Just like QuickSort.
6. Queue (8 points)

   // Add an element to the rear of this queue if there is room.
   // If there is no room left on the queue, just return.
   public void enqueue(double item) {
      // Check for room on the queue
      if (isFull) return;

      // Place the item on the queue, move the rear marker.
      queue[rear] = item;
      rear++;

      // At end of array, wrap around to the beginning
      if (rear == capacity) rear = 0;

      // increase the item count.
      numItems++;
   }

   // Remove and return the element from the front of this queue
   // If there are no elements on the queue, return 0.
   public double dequeue() {
      // Check for empty queue
      if (isEmpty) return 0.0;

      // Remove item from front of the queue, move the front marker
      double item = queue[front];
      front++;

      // At end of array, wrap around to the beginning
      if (front == capacity) front = 0;

      // Decrease the item count.
      numItems--;

      // Lastly, return the item.
      return item;
   }

   // Check if this queue is empty
   public boolean isEmpty() {
      return (numItems == 0);
   }

   // Check if this queue is full
   public boolean isFull() {
      return (numItems == capacity);
   }
}
7. Turing Machine (4 points)

\[
\cdots 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ X \ X \ X \ \cdots
\]

(a)

b) The Turing Machine adds 1 to the binary number on the tape.

8. Data Structures (3 points)

(a) Symbol Table
(b) Binary Search Tree
(c) Graph

9. True or False (6 points) Circle your answer.

T (a) P is the set of decision problems solvable in Polynomial time by a deterministic Turing Machine.

F (b) NP is the set of decision problems not solvable in Polynomial time by a deterministic Turing Machine.

F (c) For proper encapsulation, instance variables should always be declared public.

F (d) Because the Halting Problem is unsolvable, it is impossible to tell if your TSP program for Assignment 6 has an infinite loop.

T (e) A Universal Turing Machine can compute anything that any other Turing Machine could possibly compute.

F (f) If Bob wants to send a message to Alice using RSA encryption, he would first encrypt his message with his own public key, and then encrypt the result with Alice’s public key.

T (g) If P equals NP, then the Traveling Salesperson Problem can be solved in polynomial time by a deterministic Turing Machine.

F (h) If P does not equal NP, then there is no case of the Traveling Salesperson Problem for which you can find the optimal tour in polynomial time.

T (i) In a symbol table implementation using a hash table, a good hash function would distribute the keys more or less evenly over the symbol table positions.

F (j) Factoring is known to be in NP but has not been proven to be NP-complete, so the discovery of a polynomial-time algorithm for factoring would mean that P equals NP.

F (k) Factoring is known to be in NP but has not been proven to be NP-complete, so no polynomial-time algorithm for factoring is possible.

F (l) The Turing Test is a test of whether a problem can be solved by a Turing Machine.