Question 7 (a)

A concrete parse tree directly represents the grammar rules which it derives from – non-terminals become internal nodes while terminals become leaves. An abstract syntax tree simplifies a concrete parse tree in order to capture the essense/meaning behind the grammar (or in the case of a programming language, it captures the implementation). This is normally done by removing (or abstracting) out surface syntax, such as parenthese, semi-colons and braces.

In general, concrete parse trees corresponds to the programmers point of view while an abstract syntax tree corresponds to the system's point of view. The internal nodes of abstract syntax trees are usually functions which take inputs from its children, while the leaves are zero-level inputs taken directly from non-terminals.

Question 7 (b)

```
public boolean parseQuery(Scanner s){
      if(!s.hasNextPattern("SELECT"))
            return false;
      s.nextPattern("SELECT");
      if(!s.hasNextPattern("*"))
            return false;
      s.nextPattern("*");
      if(s.hasNextPattern("FROM"){
            s.nextPattern("FROM");
            if(!parseName(s))
                  return false;
      }
      if(s.hasNextPattern("WHERE"){
            s.nextPattern("WHERE");
            if(!parseName(s))
                  return false;
            if(!s.hasNextPattern("="))
                  return false;
            s.nextPattern("=");
            if(!parseData(s))
                  return false;
      }
      if(!s.hasNextPattern(";"))
            return false;
      s.nextPattern(";");
      return true;
}

public boolean parseName(Scanner s){
      if(!s.hasNextPattern("[A-Za-z]+")
            return false;
      s.nextPattern("[A-Za-z]+");
      return true;
```

```
}

public boolean parseData(Scanner s){
      if(!s.hasNextPattern("[A-Za-z0-9]*")
            return false;
      s.nextPattern("[A-Za-z0-9]*");
      return true;
}
```