1. a. Encode the following sentence in predicate logic and prove the same:
   “If all quakers are reformists and if there is a protestant who is also a quaker, then there
   must be a protestant, who is also a reformist”. June 2011 [8]
   b. Give formal definitions of terms and formulas in predicate calculus. Write the BNF
   representation of the same. June 2011 [8]
   c. Draw the parse tree for the predicate logic formula
   \((\forall x(P(x) \land Q(x)) \rightarrow (\neg P(x) \lor Q(y)))\). Which are the free and bound leaves in the

2. a. Prove the validity of the following sequents in predicate logic, where \(P, Q\) have
   arity 1:
   (i) \(\exists x(P(x) \land Q(x)) \alpha \exists xP(x) \land \exists xQ(x)\) [4]
   (ii) \(\neg \forall x \neg P(x) \alpha \exists xP(x)\) [4]
   b. Let \(\phi\) be the sentence \(\forall x \forall y \exists z(R(x, y) \rightarrow R(y, z))\), where \(R\) is a predicate symbol of
   two arguments.
   Let \(A = \{a, b, c, d\}\) and \(R^A = \{(b, c), (b, b), (b, a)\}\). Do we have \(M \models \phi\)? If
   \(A' = \{a, b, c\}\) and \(R^{A'} = \{(b, c), (a, b), (c, b)\}\). Do we have \(M' \models \phi\)? [6]
   c. Find a model such that the formulas to the left all evaluate to \(T\) but the formula to the
   right evaluates to \(F\):
   (i) \(\forall x(P(x) \rightarrow R(x)), \forall x(Q(x) \rightarrow R(x)) \alpha \exists x(P(x) \land Q(x))\)
   (ii) \(\forall x \exists y S(x, y) \alpha \exists y \forall x S(x, y)\) [6]

3. a. Construct a two process state transition system for mutual exclusion problem
   and write LTL formulas for safety, liveness, nonblocking and no-strict-sequencing
   properties. [8]
   b. What is the branching time logic? Explain LTL syntax and CTL connectives. Draw
   the parse tree for the expression: \(A[\neg X \neg p \cup E[X(p \land q) \cup \neg p]]\) [4]
   c. Prove the CTL equivalence \(A[\neg (E[\neg \phi \cup (\neg \phi \land \neg \phi)]] \cup EG \neg \phi]\) [4]
   d. With a neat diagram show that LTL is a subset of CTL*. [2]
   e. List the sub formulas of the formula \(AG(p \rightarrow A[p \land \neg A[N \cup q]]])\) [2]
4. a. (i) Beginning from state s0, unwind this system into an infinite tree and draw all possible computation paths up to length 4.
(ii) Determine whether \( M, s0 \models \Phi \) and \( M, s2 \models \Phi \) hold and justify your answer, where \( \Phi \) is the LTL or CTL formula:
\[
\neg p \rightarrow r \quad (i) \quad F t \quad (ii) \quad E G r \quad (iv) \quad E G \neg q
\]

b. Let \( M = (S, \rightarrow, L) \) be any model for CTL and let \([\Phi]\) denote the set of all \( s \in S \) such that \( M, s0 \models \Phi \). Prove the following set identities by inspecting the clauses:
\[
\begin{align*}
(a) [\top] & = S(b)[\bot] = \emptyset \cup (c) \neg \Phi = S - [\Phi](d) \Phi 1 \wedge \Phi 2 = [\Phi 1][\Phi 2] \\
(e) [\Phi 1 \vee \Phi 2] & = [\Phi 1] \neg [\Phi 2] \\
(f) [\Phi 1 \rightarrow \Phi 2] & = (S - [\Phi 1]) \neg [\Phi 2](g) [AX \Phi] = S - [EX \neg \Phi]
\end{align*}
\]

c. Find the operators to replace the ? to make the following equivalences:
\[
(a) AG(\phi \wedge \psi) = AG \phi \land AG \psi (b) EF \neg \phi = \neg ? \phi
\]

d. Express the following properties in CTL and LTL whenever possible. If neither is possible, try to express the property in CTL*:
(i) whenever \( p \) is followed by \( q \) (after infinitely many steps), then the system enters an ‘interval’ in which no \( r \) occurs until \( t \).
(ii) Event \( p \) precedes \( s \) and \( t \) on all computation paths.
(iii) After \( p, q \) is never true on all possible computation paths.
(iv) Between the events \( q \) and \( r \), event \( p \) is never true.
(v) Transitions to states satisfying \( p \) occur at most twice.
(vi) Property \( p \) is true for every second state along a path.

4. e. Find a transition system which distinguishes the following pairs of CTL* formulas:
\[
\begin{align*}
(a) AFGp, AFGq(t(b)AGFp, AGEp(c)A[p \cup r] \cup (q \cup r)].A[(p \cup q) \cup r] \\
(d) A[X p \cup XX p], AX p \cup AXAX p(c)E[GF p], EGEF p
\end{align*}
\]
f. The translation from CTL with Boolean combinations of path formulas to plain
CTL. Invent the CTL equivalents for
(a) $E(F_p \land (q \lor r))$ (b) $E(F_p \land Gq)$ (c) $E((p \lor q) \land Fp)$
(d) $A((p \lor q) \land Gp)A(Fp \rightarrow Fq)$

5. a. In what circumstances would if(B){C1} else{C2} fail to terminate.  
b. What are the general features of a framework for software verification? Write
the grammar for generating integer expression, Boolean expression and
commands. Mention various issues and concerns with the program verification
methods.

c. Write a note on Hoare’s triples. What are the provable properties of a factorial
program, in full and partial correctness? Write Hoare’s triple for the factorial
program and explain.

d. Prove that $\vdash_{\text{tot}} (x\geq 0) \text{Fac1}(y=x!)$ is valid, where Fac1 is given as below:
   
   
   ```
   y=1;
   z=0;
   while(x != z)
   {
      z = z+1;
      y= y * z;
   }
   ```

   e. For any state/store l for which l(x) = -2, l(y)=5 and l(z) = -1, determine which of
the relations below hold, justify your answer.

   (a) $l \models (x + y < z) \rightarrow \neg (x * y = z)$
   (b) $l \models \forall u(u < y) \lor (u * z < y * z)$
   (c) $l \models x + y - z < x * y * z$

   6. a. Use the proof rule for assignment and logical implication as appropriate to show
the validity of (i) $\vdash_{\text{par}} (x>0)y=x+1(y>11)$ (ii) $\vdash_{\text{par}} (T)y=x;y=x+x+y(y=3*x)$ (iii)
$\vdash_{\text{par}} (x>1)a=1;y=x;y=y-a(y>0 \land x>y)$

   b. Write down a program P such that (i) $(T)P(y=x+2)$ (ii) $(T)P(z>x+y+4)$

   c. Prove the validity of the following total-correctness sequents:

   (i) $\vdash_{\text{tot}} (x\geq 0) \text{Copy1}(x=y)$ (ii) $\vdash_{\text{tot}} (y\geq 0) \text{Multi1}(z=x*y)$
   (iii) $\vdash_{\text{tot}} (y=y0) \land (y\geq 0) \text{Multi2}(z=x*y0)$ (iv) $\vdash_{\text{tot}} (x\geq 0) \text{Downfac}(y=x!)$
   (v) $\vdash_{\text{tot}} (x\geq 0) \text{Copy2}(x=y)$, does your invariant have an active part in securing
correctness.

   (vi) $\vdash_{\text{tot}} (l(y\geq 0)) \text{Div}((x=d*y+r) \land (r<y))$

   d. Show that $\vdash_{\text{tot}} (y\geq 0) \text{Multi1}(z=x*y)$ is valid, where multi1 is:

   ```
   A=0; z=0;
   While (a != y) { z = z+x; a = a+1; }
   ```

   e. Consider methods of the form boolean certify_V(C: certificate) which return true
iff the certificate C is judged valid by the verifier V, a class in which method
certify_V resides.

   (i) Discuss how programming by contract can be used to delegate the judgment of
a certificate to another verifier.

   (ii) What potential problems do you see in this context if the resulting method-
dependency graph is circular?
f. Consider the method named `withdraw` which attempts to withdraw `amount` from an integer field `balance` of the class within which method `withdraw` lives. This method makes use of another method `isGood` which return `true` iff the value of `balance` is greater or equal to the value of `amount`.

```java
boolean withdraw(amount : int)
{
    if(amount < 0 && isGood(amount))
    {
        balance = balance - amount;
        return true;
    }
    else
    {
        return false;
    }
}
```

(i) Write a contract for the method `isGood`.

(ii) Use the contract to show the validity of the contract for `withdraw`.

Method name : withdraw
Input : amount of Type int
Assumes: 0 <= balance
Guarantees: 0 <= balance
Output : of Type Boolean
Modifies only : balance