Name: Devansh Kansara
Student Id: 21112551

Q1)

(a) Basically, in the given program, there is two conditional paths that are present. So, 4 different execution paths will be there for Prog1. They are:

1, 2, 3, 4, 9, 11, 12, 13,17 … (I) → Both the conditions are true.

1, 2, 5, 6, 7, 9, 11, 14, 15, 16, 17…. (II) → Both the conditions are false.

1, 2, 3, 4, 9, 11, 14, 15, 16, 17 … (III) → First condition is true and second is false.

1, 2, 5, 6, 7, 9, 11, 12, 13, 17 … (IV) → First condition is false and second is true.

(b)

Symbolic execution of path—I:

Edge	Symbolic State (PV)	Path Condition (PC)
1 → 2	$x \rightarrow X_0$, $y \rightarrow Y_0$	true
2 → 3	$x \rightarrow X_0$, $y \rightarrow Y_0$	$X_0 + Y_0 > 15$
3 → 4	$x \rightarrow X_0+7$, $y \rightarrow Y_0$	$X_0 + Y_0 > 15$
4 → 9	$x \rightarrow X_0+7$, $y \rightarrow Y_0-12$	$X_0 + Y_0 > 15$
9 → 11	$x \rightarrow X_0+9$, $y \rightarrow Y_0-12$	$X_0 + Y_0 > 15$
11 → 12	$x \rightarrow X_0+9$, $y \rightarrow Y_0-12$	$X_0 + Y_0 > 15$ & $2*(X_0 + Y_0 - 3 > 21)$
12 → 13	$x \rightarrow 3*(X_0+9)$, $y \rightarrow Y_0-12$	$X_0 + Y_0 > 15$ & $2*(X_0 + Y_0 - 3 > 21)$
13 → 17	$x \rightarrow 3*(X_0+9)$, $y \rightarrow 2*(Y_0-12)$	$X_0 + Y_0 > 15$ & $2*(X_0 + Y_0 - 3 > 21)$

Symbolic execution of path –II:

Edge	Symbolic State (PV)	Path Condition (PC)
1 → 2	$x \rightarrow X_0$, $y \rightarrow Y_0$	True
2 → 5	$x \rightarrow X_0$, $y \rightarrow Y_0$	$X_0 + Y_0 <= 15$
5 → 6	$x \rightarrow X_0$, $y \rightarrow Y_0$	$X_0 + Y_0 <= 15$
6 → 7	$x \rightarrow X_0$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <= 15$
7 → 9	$x \rightarrow X_0-2$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <= 15$
9 → 11	$x \rightarrow X_0$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <= 15$
11 → 14	$x \rightarrow X_0$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <=15$ & $2*(X_0 + Y_0 + 10 <= 21)$
14 → 15	$x \rightarrow X_0$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <=15$ & $2*(X_0 + Y_0 + 10 <= 21)$
15 → 16	$x \rightarrow 4*X_0$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <=15$ & $2*(X_0 + Y_0 + 10 <= 21)$
16 → 17	$x \rightarrow 4*X_0$, $y \rightarrow 3*(Y_0+10) + 4*X_0$	$X_0 + Y_0 <=15$ & $2*(X_0 + Y_0 + 10 <= 21)$

Symbolic execution for path – III:

Edge	Symbolic State (PV)	Path Condition (PC)
1 → 2	$x \rightarrow X_0$, $y \rightarrow Y_0$	true
2 → 3	$x \rightarrow X_0$, $y \rightarrow Y_0$	$X_0 + Y_0 > 15$
3 → 4	$x \rightarrow X_0+7$, $y \rightarrow Y_0$	$X_0 + Y_0 > 15$
4 → 9	$x \rightarrow X_0+7$, $y \rightarrow Y_0-12$	$X_0 + Y_0 > 15$
9 → 11	$x \rightarrow X_0+9$, $y \rightarrow Y_0-12$	$X_0 + Y_0 > 15$
11 → 14	$x \rightarrow X_0+9$, $y \rightarrow Y_0-12$	$X_0 + Y_0 > 15$ & $2*(X_0 + Y_0 - 3 <= 21)$
14 → 15	$x \rightarrow 3*(X_0+9)$, $y \rightarrow Y_0-12$	$X_0 + Y_0 > 15$ & $2*(X_0 + Y_0 - 3 <= 21)$
15 → 16	$x \rightarrow 4*(X_0+9)$, $y \rightarrow Y_0-12$	$X_0 + Y_0 > 15$ & $2*(X_0 + Y_0 - 3 <= 21)$
16 → 17	$x \rightarrow 4*(X_0+9)$, $y \rightarrow 3*(Y_0-12) + 4*(X_0+9)$	$X_0 + Y_0 > 15$ & $2*(X_0 + Y_0 - 3 <= 21)$

Symbolic execution of path – IV:

Edge	Symbolic State (PV)	Path Condition (PC)
1 → 2	$x \rightarrow X_0$, $y \rightarrow Y_0$	true
2 → 5	$x \rightarrow X_0$, $y \rightarrow Y_0$	$X_0 + Y_0 <= 15$
5 → 6	$x \rightarrow X_0$, $y \rightarrow Y_0$	$X_0 + Y_0 <= 15$
6 → 7	$x \rightarrow X_0$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <= 15$
7 → 9	$x \rightarrow X_0-2$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <= 15$
9 → 11	$x \rightarrow X_0$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <= 15$
11 → 12	$x \rightarrow X_0$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <= 15$ & $2*(X_0 + Y_0 + 10 > 21)$
12 → 13	$x \rightarrow 3*X_0$, $y \rightarrow Y_0+10$	$X_0 + Y_0 <= 15$ & $2*(X_0 + Y_0 + 10 > 21)$
13 → 17	$x \rightarrow 3*X_0$, $y \rightarrow 2*(Y_0+10)$	$X_0 + Y_0 <= 15$ & $2*(X_0 + Y_0 + 10 > 21)$

(c)

Path 1 is feasible: $X_0=10$, $Y_0=6$

Path 2 is feasible: $X_0=0$, $Y_0=-10$

Path 3 is infeasible

Path 4 is feasible: $X_0=4$, $Y_0=5$

Q2)

(a)

(a) $(\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_1 \vee \neg a_4) \wedge$.
$(\neg a_2 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_4) \wedge (\neg a_3 \vee \neg a_4)$.

(b)

(b) The given FOL is valid.

* First, we prove,

$$\forall x \exists y (P(x) \vee Q(y)) \Rightarrow (\forall x P(x)) \vee (\exists y Q(y))$$

→ We assume that the left-hand side is true. Thus, for any x, there exist at least one y such that $P(x) \vee Q(y)$ is true. For this we have to prove that right-hand side is also true.

1.) CASE (a):

→ Let there is a y_0 such that $Q(y_0)$ is true. Thus, we can say that irrespective of any x_0, we can choose y_0 such that $\exists y\, Q(y_0)$ is true and the right-hand side holds.

2) CASE (b):

→ Now, we assume that, there doesn't exist a y_0 for which $Q(y_0)$ is true. Now, as the left-hand side holds, we can

Say that $\forall x\, P(x)$ is true. Thus, the right-hand side is also true.

Hence, in both the cases, the right hand side holds true when the left hand side is true.

* Now, we prove:

$(\forall x\, P(x) \vee (\exists y\, Q(y)) \Rightarrow \forall x\, \exists y\, (P(x) \vee Q(y))$.

→ Here, we assume that the left-hand side is true.

1) CASE (a):

If $\forall x\, P(x)$ is true, then for any x_0, we can choose a y_0 such that $\exists y\, (P(x) \vee Q(y))$ is true.

2) CASE (b):

If $\exists y\, Q(y)$ is true, then for some y_0, $\exists y (P(x) \vee Q(y))$ holds for any x.

→ Thus, as both the side holds, we can say that the given FOL sentence is valid.

(c)

(066-299) WK 10 MARCH

(c). $(\forall x \exists y (P(x,y) \vee Q(x,y)) \Rightarrow ((\forall x \exists y P(x,y)) \vee (\forall x . \exists y Q(x,y)))$.

→ Consider the given example:

- Let M be a model / Universe i.e $M = \{a, b\}$. $P = \{(a,a)\}$, $Q = \{(b,b)\}$.
- For $x = a$, we can find $y = a$ such that $P(a,a)$ is true. Similarly, for $x = b$, we can find $y = b$ such that $Q(b,b)$ is true. Thus:

$\forall x \exists y (P(x,y) \vee Q(x,y))$ is true.

→ However, for the right hand side, when $x = a$, ~~we can't find~~ $Q(a,y)$ is false for every y in $\{a,b\}$, and when $x = b$, $P(b,y)$ is also false for y in $\{a,b\}$.

→ Therefore, the given FOL sentence is not universaly valid.

(d)

(d) Formula Φ:

$$\exists x \exists y \exists z \left(P(x,y) \wedge P(z,y) \wedge P(x,z) \wedge \neg P(z,x) \right)$$

(i) $M_1 \models \Phi$. Consider $x := 1$, $y := 3$, $z := 2$, then from the formula, $P(x,y)$ is true, $P(z,y)$ is true, $P(x,z)$ is true and $P(z,x)$ is false which implies that $\neg P(z,x)$ is true.

Hence $M_1 \models \Phi$.

(ii) $M_2 \not\models \Phi$. Let $x, y, z \in N$, then from $P_2(x,y)$ $\wedge$ $P(x,z)$, $y = x+1$ and $z = x+1$. Thus $y = z$. but from $P(z,y)$, $z = x$. which can't be true. So, as $P(z,y)$ is false, $M_2 \not\models \Phi$.

(iii) $M_3 \models \Phi$. Suppose $x = \{1\}$, $y = \{-1, 0, 1\}$, $z = \{0, 1\}$. and $x, y, z \in \mathcal{P}(N)$. Thus $x \subseteq y$, $z \subseteq y$, $x \subseteq z$ and $z \not\subseteq x$, $P(x,y)$ true, $P(z,y)$ true, $P(x,x)$ true but $P(z,x)$ is false.

Hence, $M_3 \not\models \Phi$.

(e)

The solution that I have given below is inspired by Will Klieber, Gihwon Kwon.

Suppose that we have a set of propositional variables X = {x_1, …, x_n}, and we desire exactly one of them to be true. In the naïve encoding, each variable must 'talk' with every other variable. That is, each variable must appear in a clause with every other variable. We can formalize this with the following functions that return a set of clauses:

$$NaïveAtLeastOne(X) = \bigvee_{i=1}^{n} x_i$$

$$NaïveAtMostOne(X) = \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} (\neg x_i \vee \neg x_j)$$

$$NaïveExactlyOne(X) = NaïveAtLeastOne(X) \wedge NaïveAtMostOne(X)$$

To reduce the number of clauses, we can divide the variables into groups. We assign a new variable, called "a commander variable", to each group. This commander variable is to be true if (at least) one of the variables in its group is true; otherwise, it is to be false. In the commander-variable method, the original variables do not need to 'talk' directly to any other variables that are not in the same group; instead, the commander variables act as proxies between original variables in different groups. To describe the commander-variable encoding more precisely, let us introduce additional notation. Let the set of propositional variables X = {x_1, …, x_n} be divided into m disjoint subsets G_1 through G_m. The commander node of group G_i is labeled "c_i". Using this notation, the logic for the commander method can be encoded in CNF form as follows:

1. **At most one variable in a group can be true:** For each group G_i, we encode the following clauses:

$$\bigwedge_{x_j \in G_i} \bigwedge_{x_k \in G_i, k<j} (\neg x_j \vee \neg x_k)$$

2. **If the commander variable of a group is false, then none of the variables in the group can be true:** For each group G_i, we encode the following clauses:

$$\bigwedge_{x_j \in G_i} (c_i \vee \neg x_j)$$

3. **Exactly one of the commander variables is true:** This can be encoded either by the pair-wise method or by a recursive application of the commander method.

Let's consider how many clauses are required for each group. Note that Constraint 1 above requires n∗(n-1)/2, where n is the number of variables in the group. Constraint 2 requires 1 clause. Thus, the total number of clauses per group is (n∗(n+1)/2) + 1.

However, to analyze the performance of the commander approach (in terms of many clauses and extra variables are required), let us consider a grouping method wherein the variables are, at each stage of the hierarchy, divided into groups of k variables. For example, k=2, we get binary tree shown in Fig. 1. From the diagram, it is clear that the number of groups, in the asymptotic limit of a large number of variables n, is:

$$\begin{aligned}
&\sum_{i=1}^{\infty}(n/k) = (n/k) + (n/k^2) + (n/k^3) + \cdots \\
&= (1/k + 1/k^2 + n/k^3 + \cdots)n \\
&= \left(\frac{1}{1-1/k} - 1\right)n \\
&= \left(\frac{1/k}{1-1/k}\right)n
\end{aligned}$$

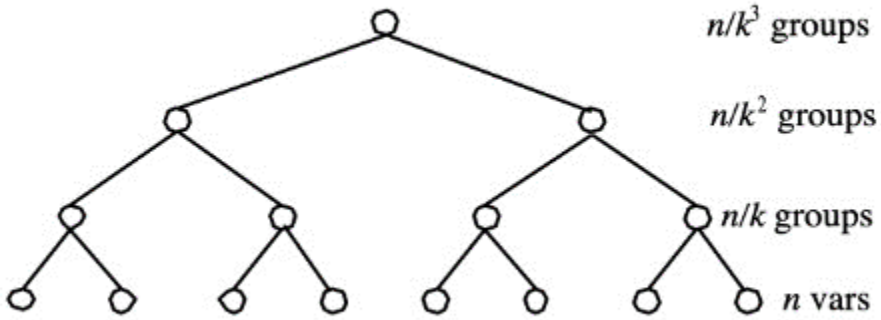

Figure 1: Binary tree of groups

The number of additional variables is equal to the number of groups. To get the number of clauses, we multiply this by the number of clauses per group, getting a result of:

$$\left(\frac{(k+1)/2+(1/k)}{1-1/k}\right)n$$

We find that k = 3 is the best choice for reducing the number of clauses. In this case, there are n/2 more variables in addition to the total of 3n clauses. This encoding therefore employs a maximum of O(n) clauses and variables.

Q3)

(a)

(068-297) WK 10 MARCH

(Q-3)

(1) Let $I(x)$ be a predicate that returns true if x is an integer and false otherwise.

To express quantifier free constraints in FOL:

$$\bigwedge_{1 \le r_1, r_2, c_1, c_2 \le n \atop r_1 \ne r_2 \text{ or } c_1 \ne c_2} p_{r_1 c_1} \ne p_{r_2 c_2}.$$

$$\bigwedge_{1 \le r, c \le n} I(p_{rc})$$

$$\bigwedge_{1 \le r, c \le n} 1 \le p_{rc} \le n^2$$

$$\bigwedge_{1 \le r \le n} \sum_{1 \le c \le n} a_{rc} = \frac{n^2(n^2+1)}{2}.$$

$$\bigwedge_{1 \le c \le n} \sum_{1 \le r \le n} a_{rc} = \frac{n^2(n^2+1)}{2}$$

$$\bigwedge_{1 \le r \le n} a_{r, n-r+1} = \frac{(1+n^2) \cdot n^2}{2}$$

FEB - 2018

Q4)

(d)

sym.py:322 is not covered because we did not execute the file as the main program.
sym.py: 132 is not covered because **WHILE** only has <= / < / = / >= / > relational operators only.
sym.py: 151 is not covered because WHILE only has not | and | or logical operators only.
sym.py: 173 is not covered because WHILE only has + | - | * | / arithmetic operators only.

(e)

The symbolic execution engine diverges for the following program:

havoc x, y; while x > 0 do {while y > x do {y: = y / 2 - 1}; x: = x / 2 - 1}