1.) Recall the HALTING problem which takes a program and a string as input, and consider the following variant thereof:

**HALTING-X**

INSTANCE: Two programs $\Pi_1, \Pi_2$ that take a string as input.

QUESTION: Does there exist at least one input string $I$ such that both $\Pi_1$ and $\Pi_2$ halt on $I$.

(a) The following function $f$ provides a polynomial-time many-one reduction from HALTING to HALTING-X: for a program $\Pi$ and a string $I$ let $f(\Pi, I) = (\Pi_1, \Pi_2)$ with

\[
\begin{align*}
\Pi_1(\text{string } S) &= \text{return;} \\
\Pi_2(\text{string } S) &= \text{if } (S = I) \{ \text{call } \Pi(S) \} \text{ else } \{ \text{while(true)} \}; \text{ return;}
\end{align*}
\]

Show that $(\Pi, I)$ is a yes-instance of HALTING $\iff$ $(\Pi_1, \Pi_2)$ is a yes-instance of HALTING-X.

(9 points)

(b) Tick the correct statements (for ticking a correct statement a certain number of points is given; ticking an incorrect statement results in a substraction of the same amount; you cannot go below 0 points):

- Since HALTING is decidable, our reduction from (a) shows that HALTING-X is decidable.
- Since HALTING is undecidable, our reduction from (a) shows that HALTING-X is undecidable.
- Since HALTING is semi-decidable, our reduction from (a) shows that HALTING-X is semi-decidable.
- Since HALTING is not semi-decidable, our reduction from (a) shows that HALTING-X is not semi-decidable.
- A reduction from HALTING-X to HALTING would show that HALTING-X is semi-decidable.
- A reduction from HALTING-X to HALTING would show that HALTING-X is undecidable.

(6 points)
2.) (a) Consider the clauses $C_1, \ldots, C_5$ in dimacs format (in this order, shown in the box) which are given as input to a SAT solver. Apply CDCL to solve the CNF using the convention that if a variable is assigned as a decision, then it is assigned 'false'. Further, select variable 2 as the first decision variable that is assigned.

- Each time when a conflict occurs and after backtracking, draw the implication graph and indicate all UIPs and mark the first UIP. For the first UIP, indicate its asserting conflict clause.
- Is the given CNF satisfiable, unsatisfiable, or valid? Justify your answer.

(b) Consider the theory $T_A$ of arrays and the following formula

$$\varphi: \ i_1 \neq j \lor a[j] \neq v_1 \lor i_1 \neq i_2 \lor a<i_1 \triangleleft v_1>(i_2 \triangleleft v_2)[j] \triangleq a[j].$$

If $\varphi$ is $T_A$-valid, then provide a proof in the semantic argument method (similarly to the proofs in the lecture and on the extra sheets). If $\varphi$ is not $T_A$-valid, then provide a counter-example.

Besides the equality axioms reflexivity, symmetry and transitivity, you have the following ones for arrays:

- $\forall a, i, j \ (i \triangleright j \rightarrow a[i] = a[j])$ (array congruence)
- $\forall a, v, i, j \ (i \triangleright j \rightarrow a(i \triangleleft v)[j] \triangleq v)$ (read-over-write 1)
- $\forall a, v, i, j \ (i \neq j \rightarrow a(i \triangleleft v)[j] \triangleq a[j])$ (read-over-write 2)

Please be precise. In a proof indicate exactly why proof lines follow from some other(s) and name the used rule. If you use derived rules you have to prove them. (11 points)
3.) (a) Let $p$ be the following program:

$$
\begin{align*}
&x := 1; y := 1; z := 0; \\
&w hile \ x < n \ do \\
&\quad x := x + 1; \\
&\quad z := z + 6; \\
&\quad y := y + z \\
&od
\end{align*}
$$

Give a loop invariant and variant for the while loop in $p$ and prove the validity of the total correctness triple $[n > 1] p [2 * y = n * z + 2]$. 

Hint: Make sure that your invariant expresses equalities among $y, z, x$, as well as equalities among $z, x$.

(10 points)

(b) Provide a non-trivial pre-condition $A$ and a non-trivial post-condition $B$, such that the total correctness triple $[A] p [B]$ is valid. Trivial means equivalent to true or false, so your precondition $A$ and postcondition $B$ should not be equivalent to true or false. The program $p$ is given below.

Program $p$:

\[ \text{if } x \neq 0 \text{ then skip else abort} \]

(2 points)

(c) Consider the following partial correctness triple:

\[ \{x = y\} x := y + 1; y := x + 1 \{x = y - 1\} \]

Is the above Hoare triple valid? If so, give a formal proof. Otherwise, give a counterexample.

(3 points)
4.) (a) Provide a non-empty simulation relation $H$ that witnesses $M_1 \leq M_2$, where $M_1$ and $M_2$ are shown below. The initial state of $M_1$ is $s_0$, the initial state of $M_2$ is $t_0$:

Kripke structure $M_1$:

Kripke structure $M_2$:
(b) Consider the following Kripke structure $M$:

For each of the following formulae $\varphi$,

i. indicate whether the formula is in CTL, LTL, and/or CTL*, and

ii. list the states $s_i$ on which the formula $\varphi$ holds; i.e. for which states $s_i$ do we have $M, s_i \models \varphi$?

(If $\varphi$ is a path formula, list the states $s_i$ such that $M, s_i \models A \varphi$.)

<table>
<thead>
<tr>
<th>$\varphi$</th>
<th>CTL</th>
<th>LTL</th>
<th>CTL*</th>
<th>States $s_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x U y$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F(x \land z)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$EF(y)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$EG(z)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AX(z)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5 points)
(c) **LTL tautologies**

Prove that the following formulas are tautologies, i.e., they hold for every Kripke structure $M$ and every path $\pi$ in $M$, or find a Kripke structure $M$ and path $\pi$ in $M$, for which the formula does not hold and justify your answer.

i. $F(y \land Gx) \Rightarrow (y \land ((Xy) \ U (Gx)))$

ii. $(y \land ((Xy) \ U (Gx))) \Rightarrow F(y \land Gx)$

(6 points)