Note

• This homework is more like a take-home exam.  
  Your solution should meet the requirements of exam 1 solution.  
  It will be graded in the same way.  
  It will have much higher weight than other homeworks.

• You cannot ask questions about how to proceed, whether you are on the right track, etc.  
  You can “translate” such questions to exam 1 solution, and we will answer those.

• You can do this homework individually or with one partner. It’s entirely up to you.  
  Can’t find a partner? Not happy with your partner? Want to leave your partner? Not my concern.

• Your solution should be neat and readable.
Problem 1 [30 points]

This program below is a variation of the Otway-Reese protocol. It has an attacker, kdc Z, client A and server B. The attacker can read-write the channel and get A’s old password (only when A is between sessions).

```plaintext
Protocol(Z, A, B) { // kdc, client, server
    chan ← []; // connect history
    hst ← []; // connect history
    kAZ ← random(); // initial A-Z key
    kBZ ← random(); // initial B-Z key
    startSystem(Attacker());
    startSystem(Kdc(Z,A,B,kAZ,kBZ));
    startSystem(Client(A,Z,B,kAZ));
    startSystem(Server(B,Z,A,kBZ));
}

Attacker() { // initially has A, B, Z, all programs
    α; // functions executable by attacker
    function rChan (α -- chan;) // read chan
    function wChan(x) (chan ← x;) // write chan
    function getPwdA() { // get A.key iff A.t at 1
        if (A.t at 1) {
            α.append(A.key);
            A.key ← Z.keyA ← random();
        }
    }
}

Kdc(Z, A, B, kAZ, kBZ) { // atomicity points: 1
    keyA ← kAZ;
    keyB ← kBZ;
    t ← startThread(client());
    return;
    function kdc() {
        while (true) {
            1: msg ← rx([B,Z,..]);
                x ← dec(keyB, msg[2]);
            if (x.size = 4 and x[0,1] = [A,B]) {
                nB ← x[2];
                y ← dec(keyA, x[3]);
            } if (y.size = 3 and y[0,1] = [A,B]) {
                nA ← y[2];
                kAB ← random();
                rA ← enc(keyA, [nA, kAB]);
                rB ← enc(keyB, [nB, kAB]);
                tx([Z,B,rA,rB]);
            }
        }
    }
}
Client(A, Z, B, kAZ) { // atomicity points: 1, 2
    key ← kAZ;
    t -- startThread(client());
    return;
    function client() {
        while (true) {
            1: nL ← random();
                tx([A,B,1, enc(key, [A,B,nL])]);
                msg ← rx([B,A,..]);
                x ← dec(key, msg[2]);
            if (x.size = 2 and x[0] = nL) {
                kAB ← x[1];
                hst.append([A,kAB]);
                tx([A,B,2, enc(kAB, 'HELLO')]);
            }
        }
    }
}
Server(B, Z, A, kBZ) { // atomicity points: 1,2,3
    key ← kBZ;
    t -- startThread(server());
    return;
    function server() {
        while (true) {
            1: msg ← rx([A,B,1,..]);
                nL ← random();
                tx([B,Z, enc(key, [A,B,nL, msg[3]])]);
                msg ← rx([Z,B,..]);
                x ← dec(key, msg[2]);
            if (x.size = 2 and x[0] = nL) {
                kAB ← x[1];
                hst.append([B,1, kAB]);
                tx([B,A, msg[2]]);
            } 2: msg ← rx([A,B,2,..]);
                if (dec(kAB, msg[3]) = 'HELLO') {
                    hst.append([B,2, kAB]);
                }
        }
    }
}
```
Problem 1 (cont)

Part a.

Does $\text{Inv} A_1$ hold, where

$$A_1 : ((j \in \text{hst.keys}) \land j > 0 \land \text{hst}[j] = [A,p]) \Rightarrow \text{hst}[j-1] = [B,1,p]$$

If yes, assume that $A$ appends $[A,p]$ to $\text{hst}$ at time $t_0$ and prove that $[B,1,p]$ is the last entry in $\text{hst}$ just before $t_0$.

If no, come up with a counter-example evolution, i.e., ending in a state where $A_1$ does not hold.

Part b.

Does $\text{Inv} A_2$ hold, where

$$A_2 : ((j \in \text{hst.keys}) \land j > 0 \land \text{hst}[j] = [B,2,p]) \Rightarrow \text{hst}[j-1] = [A,p]$$

If yes, assume that $B$ appends $[B,2,p]$ to $\text{hst}$ at time $t_0$ and prove that $[A,p]$ is the last entry in $\text{hst}$ just before $t_0$.

If no, come up with a counter-example evolution, i.e., ending in a state where $A_1$ does not hold.

(Hint: $\text{Inv} \psi(A,\text{key})$ may hold. $\text{Inv} \psi(B,\text{key})$ may not hold.)

Problem 2 [30 points]

Repeat problem 1 after changing the kdc-to-server message to include the response to $A$ inside the response to $B$.

The change can be made as follows:

- In function kdc

  \[
  \text{rA} \leftarrow \text{enc(keyA, [nA,kAB]);}
  \text{rB} \leftarrow \text{enc(keyB, [nB,kAB]);}
  \text{tx([Z,B,rA,rB]);}
  \]

  becomes

  \[
  \text{rA} \leftarrow \text{enc(keyA, [nA,kAB]);}
  \text{rB} \leftarrow \text{enc(keyB, [nB,kAB,rA]);}
  \text{tx([Z,B,rB]);}
  \]

- In function server

  \[
  \text{msg} \leftarrow \text{rx([Z,B,...]);}
  \text{x} \leftarrow \text{dec(key,msg[3]);}
  \text{if (x.size = 2 and x[0] = nL) {}
  \text{tx([B,A,msg[2]]);}
  \}

  becomes

  \[
  \text{msg} \leftarrow \text{rx([Z,B,...]);}
  \text{x} \leftarrow \text{dec(key,msg[3]);}
  \text{if (x.size = 3 and x[0] = nL) {}
  \text{tx([B,A,x[2]]);}
  \}