```
function Equiv(a,b) return ((a and b) or ((not a) and (not b))); end
         function Xor(a,b) return ((a or b) and (not (a and b))); end
         function Nand(a,b) return (not (a and b)); end
         function ComputeRows(header) return 2<sup>h</sup>eaderend
         function Split(s, delimiter) local result = ; for match in (s..delimiter):gmatch("(.-)"..delimiter)
do table.insert(result, match); end return result; end
         function EvaluateFormula(formula)
        local parsedFormula = "function res() return(" .. string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(string.gsub(stri
"",""),"\dot{\dot{\xi}}\ddot{\dot{\xi}}","Impl"),"_{\epsilon,\epsilon Equiv\epsilon),\epsilon <>\epsilon,\epsilon Equiv\epsilon),\epsilon}
         chunk = load(parsedFormula); chunk(); local result = res(); return result; end
         function to
Bits(num) local t="" – will contain the bits while num;
0 do local rest =
\operatorname{math.fmod}(\operatorname{num}, 2) if (\operatorname{rest} == 1) then t = "1" .. t else t = "0" .. t end
         num = (num-rest)/2 end return t; end
         function printTruthValue(expr, dTrue, dFalse)
         local returnVal = ""
         if (expr) then returnVal = dTrue; else returnVal = dFalse; end
         return returnVal; end
         function parse(commaSepVariables, commaSepDisplayVariables, commaSepResultRows, com-
maSepResultDisplayRows, displayTrue, displayFalse)
         print("v0.0.2")
         local vrbls = Split(commaSepVariables, ","); local numberOfColumns = (vrbls); local rows
= ComputeRows(numberOfColumns); local dVrbls = Split(commaSepDisplayVariables, ",");
local resRows = Split(commaSepResultRows, ","); local dResRows = Split(commaSepResultDisplayRows,
",");
        local\ dHeader = string.gsub(commaSepDisplayVariables,",","") .. "" .. string.gsub(commaSepResulting) .. "" " .. string.gsub(commaSepResulting) .. " " .. string.gsub(commaSepResult
",", " ") .. "
hline";
         if ((dVrbls) = (vrbls)) then print("Error: The number of variables does not match the
number of display variables."); return end
         if ((dResRows) = (resRows)) then print("Error: The number of statements does not match
the number of display statements."); return end
         local tableContent = dHeader;
         for i = (rows - 1), 0, -1 do local bitString = toBits(i);
         while bitString; numberOfColumns do bitString = "0".. bitString end
         local wVrbls = commaSepVariables; local wCommaSepRows = commaSepResultRows for
ii = 1,numberOfColumns do wVrbls = string.gsub(wVrbls, vrbls[ii], (string.sub(bitString,ii,ii)
== "1" ) and "+" or "-" ) wCommaSepRows = string.gsub(wCommaSepRows, vrbls[ii],
(string.sub(bitString,ii,ii) == "1") and "+" or "-") end
        local aWVrbls = Split(string.gsub(string.gsub(wVrbls, "+", "true"),"-", "false"), ",");
         local aWCommaSepRows = Split(string.gsub(string.gsub(wCommaSepRows, "+", "true"),"-
", "false"), ",");
         local row = "";
         for c = 1, (aWVrbls) do row = row .. printTruthValue(EvaluateFormula(aWVrbls[c]), dis-
playTrue, displayFalse) .. " "; end
         for c = 1, (aWCommaSepRows) do row = row .. printTruthValue(EvaluateFormula(aWCommaSepRows)
```

function Impl(a,b) return (not a or b); end

i)
$$\neg (Z \land Z)$$
.

iv)
$$P \vee (P \wedge Q)$$
.

vii)
$$\neg W \wedge W$$
.

ii)
$$P \rightarrow P$$
.

v)
$$B \rightarrow (P \rightarrow B)$$

v)
$$B \to (P \to B)$$
. viii) $(R \to A) \to (\neg A \to \neg R)$.

iii)
$$P \wedge (P \vee Q)$$
.

vi)
$$(\neg S \land ((\neg S) \lor Q)) \leftrightarrow S$$
. ix) $K \land B$.

ix)
$$K \wedge B$$

Ei. 2 (1 pt) Escribe fórmulas lógicas (de primer orden) que, a tu criterio, capturen mejor cada una de las siguientes afirmaciones.

- i) Cada persona viva respira.
- iv) No existen estudiantes en Ciudad Universitaria que sean felices.
- ii) 2 es el único primo par.
- v) Todos los peces del acuario de la facultad se aparean con
- iii) Existe un hombre inmortal.
- un individuo.

Ej. 3 (1 pt) Escribe la negación de las siguientes proposiciones. Si el inciso está en español, da tu respuesta también en español.

i)
$$P \leftrightarrow Q$$
.

i)
$$P \leftrightarrow Q$$
. iv) $\exists x (P(x) \land (Q(x) \land S(x)))$.

vii) Si *x* es un número primo y es mayor que 4, n es impar.

ii)
$$\neg P \rightarrow A$$

ii)
$$\neg P \rightarrow A$$
. v) $\forall a(P(a) \rightarrow \exists b(R(a,b)))$.

iii)
$$R \rightarrow (P \rightarrow R)$$

iii)
$$B \to (P \to B)$$
. vi) $\exists b \forall x (\forall y (P(y)) \leftrightarrow (Q(x, y) \land R(b)))$

viii) Hay cierto elemento de A que es real, pero no real.

Ej. 4 (1 pt) Sea $A := \{x \in \mathbb{R} : x > 0\}$. Para cada $y \in \mathbb{R}$ se define el conjunto B(y) (pues depende de y), como $B(y) := \{x \in \mathbb{R} : |x - y| < 0.15\}$. Utilizando operaciones de conjuntos, escribe en términos de los conjuntos anteriores, la colección cuyos elementos sean:

- i) Todos los enteros menores o iguales a 0.
- ii) Los reales negativos mayores a -0.15.
- iii) Todos los irracionales cuya distancia a 2 es mayor o igual a 0.15.
- iv) Todos los racionales que distan de algún entero en menos de 0.15.

Holaa