# Birla Institute of Technology and Science Pilani, Hyderabad Campus
# CS F214 Logic in Computer Science, I Semester 2021-2022
# (Programming) Assignment-3
Total Weightage: 15% of final grade

## Preface

In this preface, we first informally describe the problem outlining some requisite background and then list instructions you must follow.
After this, the problem statements for the three parts of the assignment are described in detail.

**Brief outline of the problem**

This coding assignment consists of 3 parts. Part A consists of parsing the Horn Formula given in a text file. In Part B, you are expected to to write code to check the satisfiability of the Horn Formula parsed in Part A. In Part C you will find out a possible satisfying assignment of the propositional atoms in case the Horn Formula parsed in Part A is satisfiable and print "The given horn formula is not satisfiable" otherwise.

**Definition of a Horn Formula**

Horn formulas are conjunctions of Horn clauses. Horn clauses are an implication whose hypothesis (left side of the arrow) A is a conjunction of the proposition of type P and whose conclusion (right side of the arrow) is of type P (P::= $\bot$ | $\top$| atom) also. Here, an example of a Horn formula is shown, having conjunctions of Horn clauses.

$H = (p \rightarrow q) \wedge (t \wedge r \rightarrow \top) \wedge (p \wedge r \wedge s \rightarrow \bot)$

A Horn formula is a formula φ of propositional logic if it can be generated as an instance of H in this grammar:

$P ::= \bot | \top | p$

$A ::= P \mid P \wedge A$

$C ::= A \rightarrow P$

$H ::= C \mid C \wedge H$

**Definition of Satisfiability**

A formula φ in propositional logic is said to be satisfiable if it has a valuation in which it evaluates to T.

**General Instructions (applicable to all parts):**

- You are only allowed to use the C Programming language. Your file should compile with gcc and run on a unix/linux system. One option is to use the onlinegdb website, which uses gcc 9.3.0 c99. You must ensure that your code compiles there.

- Please work as a team. **There should be only one submission per team, which is a single C file of the form TeamX_PartY.c** where X is your team number as entered in the Google Sheet that was shared previously and Y is either A, B or C referring to whether the submission is that of Part A, B or C.
- The first few lines of your C file should have as comments your team number and the names and ID numbers of all the team members.
- Code corresponding to Part A, B and C should be submitted separately on the corresponding CMS pages. Each part has a separate submission deadline, as noted in the following pages.
- You will not be able to submit after the deadline. Submit what you have, to possibly receive partial credit. No email submissions will be accepted.
  **Plan your time well and submit before 11pm.**
- If you fail to submit Part A, you can still submit Part B on the subsequent day and receive credit for Part B, but you won't receive marks for Part A. Likewise for Part C. If you didn't submit Part A, you are **not** allowed to start with some other team's Part A code, etc.
- Do not share code with other teams. Copied code will be awarded zero marks for the entire assignment. Expecting all of you to be honest and take pride in your own work.

# Assignment 3, Part A

## Weightage: 5% of final grade

## Deadline: Fri, Nov 26, 2021, 11pm

Implement a C program to parse a Horn formula given in the file that is passed as a command line argument. The parsed formula should then be printed. The propositional atoms p1,p2,... are represented as 1,2,.....; and F and T are used to represent $\perp$ and T respectively in the input file.

**Input format :**

Line 1 : Number of unique propositional atoms n

Line 2 : Number of horn clauses m

The following m lines represent m horn clauses which will be joined by conjunction

**Constraints :**

- $1 \leq n \leq 9$
- $1 \leq m \leq 100$

**Output format :**

Print the parsed horn formula as shown in the example below.

**Example:**

Eg1: ***File Input:***

4

3

1^2^3>1

2^4>1

1^3>3

***Expected Output:*** (p1^p2^p3>p1)^(p2^p4>p1)^(p1^p3>p3)

**Function Template:**

//save the following file as TeamX_PartA.c and Include the necessary libraries, you may use the structure shown below

// **parse_input** reads the input file and initializes the horn formula.

```
#define INPUT_FILE “input.txt”

typedef struct horn {
   int noc;       //number of horn clauses in the horn formula
   int p;         //number of unique propositional atoms in the horn formula
```

```
    int* nt;        //array representing the length of lhs of the implicants in each horn clause
    int** lhs;      //array representing the lhs of the implicants in each horn clause
    int* rhs;       //array representing the rhs of the implicants in each horn clause
} horn;

horn* parse_input (){
}
```

# Assignment 3, Part B

## Weightage: 5% of final grade

## Deadline: Mon, Nov 29, 2021, 11pm

Parse the Horn formula using the functions implemented in Part A. After parsing the Horn formula, determine if it is satisfiable or not.

**Input format (identical to Part A) :**

Line 1 : Number of unique propositional atoms n

Line 2 : Number of horn clauses m

The following m lines represent m horn clauses which will be joined by conjunction

**Constraints :**

- $1 \leq n \leq 9$
- $1 \leq m \leq 100$

**Output format :**

The given horn formula is satisfiable (if the horn formula is satisfiable)

The given horn formula is not satisfiable (if the horn formula is not satisfiable)

Example : **Input:**

4

3

1^2^3>1

2^4>1

1^3>3

**Expected Output:** The given horn formula is satisfiable

**<u>Function Template:</u>**

```
//save the following file as TeamX_PartB.c
//Include the necessary libraries and use the function defined in PartA
//HORN_SAT returns 1 if the horn formula is satisfiable and 0 otherwise.
int HORN_SAT(horn* h){
}
```

# Assignment 3, Part C

## Weightage: 5% of final grade

## Deadline: Tue, Nov 30, 2021, 11pm

Parse the Horn formula using the functions implemented in Part A. Using the function implemented in Part B, check if the Horn formula is satisfiable or not. If the Horn formula is satisfiable, print one possible assignment for the formula which makes it satisfiable. Otherwise, print "The given horn formula is not satisfiable".

**Input format (identical to Parts A and B) :**

Line 1 : Number of unique propositional atoms n

Line 2 : Number of horn clauses m

The following m lines represent m horn clauses which will be joined by conjunction

**Constraints :**

- $1 \leq n \leq 9$
- $1 \leq m \leq 100$

**Output format :**

*n* space separated T/F indicating the assignment of the n atoms such that the $i^{th}$T/F indicates the assignment of p_i in the horn formula.

**Example:**

Eg1: **Input:**

4

3

1^2^3>1

2^4>1

1^3>3

**Expected Output:** F F F F

(Note: More than one satisfying assignments may be possible, you can print any one satisfying assignment)

**<u>Function Template:</u>**

//save the following file as TeamX_PartC.c
//Include the necessary libraries and use the functions defined in PartA and PartB

/***HORN_assignment** prints a satisfiable assignment if the horn formula is satisfiable and prints “The given horn formula is not satisfiable otherwise ***/

```
void HORN_assignment(horn* h){
}
```