**TOC CASE STUDY**

GROUP NO: 9

TOPIC: CALCULATOR

|  |  |  |
| --- | --- | --- |
| S.NO | NAME OF THE STUDENT | ROLL NUMBER |
| 1 | KSSH.HARSHITHA | CB.EN.U4CSE21427 |
| 2 | KHAJI MAHAMMAD SAMI | CB.EN.U4CSE21430 |
| 3 | K. ADITHYA REDDY | CB.EN.U4CSE21431 |
| 4 | K. AKHILA PAVANI | CB.EN.U4CSE21433 |
| 5 | UJWAL SRIMANTH | CB.EN.U4CSE2140 |

PROBLEM STATEMENT**:**

Implementation of the Turing machine and NPDA of the Calculator which deals with the mathematical expressions

**OVERVIEW:**

Calculator is a tool which makes calculations of the complex and long expressions easy. We can get the answer within no time so it saves our time.

Our calculator deals with the addition, subtraction,

Multiplication, division of the numbers. Input will be in the form of expressions with all types of operators in between say like 2+3-1.

Now, we can see how to solve the problem using the NPDA and Turing Machine. For that, we need to know what are they in the first place. Let us get to know them!

**CONSTRAINTS:**

* Expression will have maximum of 3 operators
* NPDA has no multiplication and division operators
* If the result is going to be in negative then no further operators and operands
* inputs are always having exactly three distinct operators.
* the operator inputs are always in the form of +,-,\*,/.

the number inputs are given in unary format.

* the extra memory needed to store the result after division and multiplication is given by the user correctly.
* the expression start is given by P and is Given by T.

some extra temporary alphabets inside Turing machine to make it easier to implement or draw.

* the integer given are always positive in nature.
* + is represented by A , - by 0 , \* by Y , / by C and extra memory for division is represented by Y and extra memory for multiplication is given by M.
* If 11\*111(2\*3) is performed and memory input is not exactly 6 bits then the Turing machine halts
* Complex cases such as 0/0, infinity\*0 , 2/0 are ignored .
* As zero is increasing the states It is also ignored so numbers can just be positive integers
* We have assumed to do single tapee Tuning machin
* Our Turing machine is a transducer which leaves a output in the Turing machine.
* The initial states are q17 and the accepting states are q61, q101, q99, q64.
* We have ignored brackets to make implementation simpler
* We have taken the numbers from 0-5 to reduce the complexity.

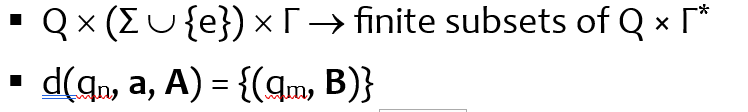
WHAT IS A NON-DETERMINISTIC AUTOMATA- THE NPDA?

An NPDA-Non-Deterministic Pushdown Automaton is an extended version of PDA with non-deterministic behavior.

It basically comprises of non-finite automaton and a stack, which work together to store and retrieve symbols during computation.

An NPDA can be representedas 

* + Q – All states present in the automata.
  +  is the representation of input alphabets.
  + represents thee alphabet set stack alphabets.
  + d is the transition function from



* + q0 is the start state(q0 ∈ Q)
  + z is the initial stack symbol (z ∈ Γ)

F is a representation of all collection of final states

**WHY WE CHOOSE NPDA OVER DPDA :**Because **NPDA** allows for multiple transitions from a given state with a particular input symbol and it uses a stack to store and manipulate symbols.

* NPDA can branch into multiple paths simultaneously, and explores different possibilities.
* NPDA accepts a string if there exists at least one computation path that leads to an accepting state.

* **DPDA** allows only one unique transition from a given state with a particular input symbol. Like NPDA, it also uses a stack to store and manipulate symbols , it follows a deterministic behavior meaning it always knows which transition to take for a given input symbol in a particular state.
* DPDA accepts a string if there exists exactly one computation path that leads to an accepting state.

So, here in the calculator we choose NPDA because in a given state for a particular input there are multiple transitions and we cannot show output without multiple transitions which is not possible in DPDA

**DESIGN EXPLANATION FOR NPDA**

NPDA is defined by the set of 6 tuples where

 all possible internal statess

Here in our NPDA we have the following internal states{q6,q7,q10,q11,q15,q16,q17,q18,q20,q21,q22,q23,q26,q27,q28,q29,q30,q31 }

* Sigma here represents the alphabet set we use as input in our NPDA. Here sigma consists of the following symbols

{0,1,2,3,4,5}

* The Greek symbol gamma represents the set of alphabets present in the stack memory.
* δ represents the transition function. For example δ(q1,x,Z)=(q2,y) where q1 is thee current state , x is thee input symbolize is thee current topp most element in the stack, q2 is the next state, y is the current top most element in the stack.
* q0 represents the starting state of the NPDA and qf represents the final state of the NPDA

here the output is in the form of unary when we enter the digits. Finally there will be a result so it act as a transducer in our problem statement so its not about some particu

**TRANSITIONS FOR NPDA:**

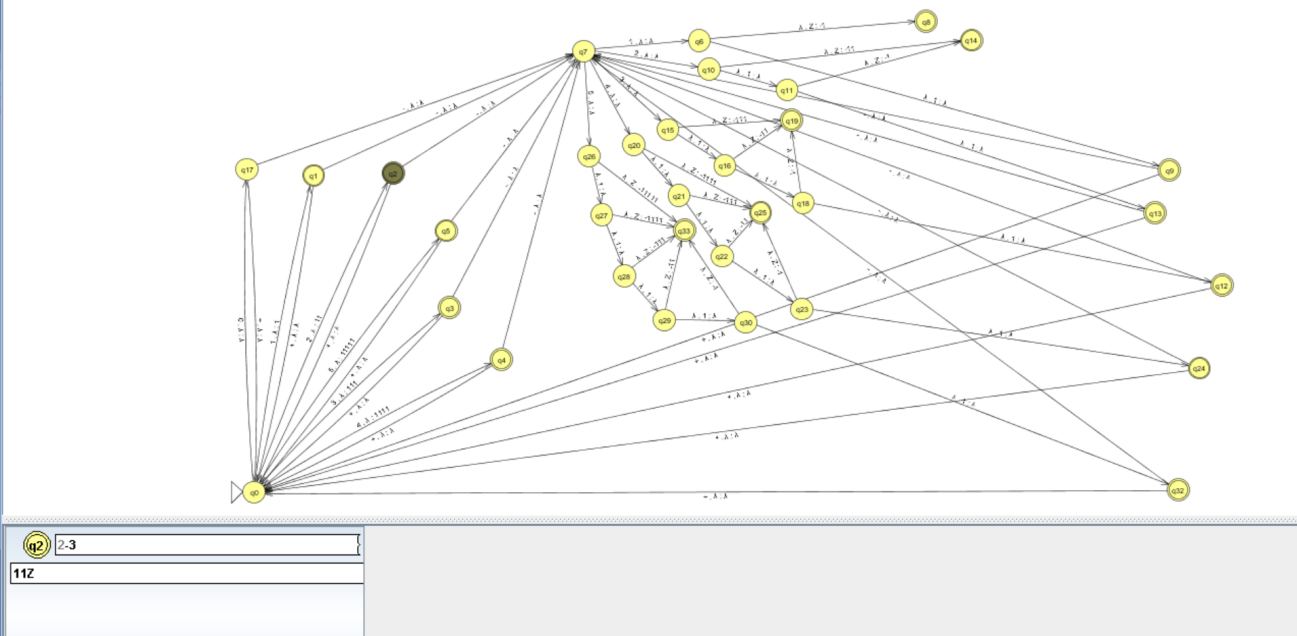
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Current State | Input Symbol | Top of the stack | operation | Next state | Current top of the stack |
| q0 | 0 | Z | No push/pop | q17 | Z |
| q0 | 1 | Z | push | q1 | 1Z |
| q0 | 2 | Z | Push | q2 | 11Z |
| q0 | 3 | Z | push | q3 | 111Z |
| q0 | 4 | Z | Push | q4 | 11114 |
| q0 | 5 | Z | Push | q5 | 111115 |
| q17 | - | z | No push/pop | Q7 | Z |
| Q1 | - | 1 | No push/pop | Q7 | 1Z |
| Q2 | - | 1 | No push/pop | Q7 | 11Z |
| q3 | - | 1 | No push/pop | Q7 | 111Z |
| Q5 | - | 1 | No push/pop | Q7 | 1111Z |
| Q4 | - | 1 | No push/pop | Q7 | 1111Z |
| Q7 | 1 | 1 | No push/pop | Q6 | 1 |
| Q7 | 2 | 1 | No push/pop | Q10 | 1 |
| Q7 | 3 | 1 | No push/pop | Q15 | 1 |
| Q7 | 4 | 1 | No push/pop | Q20 | 1 |
| Q7 | 5 | 1 | No push/pop | Q26 | 1 |
| Q6 | λ | Z | pop | q10 | -1 |
| Q10 | λ | 1 | pop | Q11 | 1 |
| Q15 | λ | 1 | pop | Q16 | 1 |
| Q20 | λ | 1 | pop | Q21 | 1 |
| Q26 | λ | 1 | pop | Q9 | 1 |
| Q11 | λ | 1 | pop | Q14 | z |
| Q10 | λ | Z | pop | Q14 | -11 |
| Q15 | λ | Z | Pop | Q19 | -111 |
| Q20 | λ | Z | Pop | Q25 | -1111 |
| Q26 | λ | z | pop | Q33 | -11111 |
| Q16 | λ | 1 | Pop | Q18 | 1 |
| Q21 | λ | 1 | Pop | Q22 | 1 |
| Q27 | λ | 1 | pop | Q28 | 1 |
| Q6 | λ | 1 | Pop | Q9 | Z |
| Q11 | λ | 1 | Pop | Q13 | Z |
| Q18 | λ | 1 | Pop | Q12 | Z |
| Q23 | λ | 1 | Pop | Q24 | Z |
| Q30 | λ | 1 | pop | Q32 | z |
| Q13 | + | Z | No push/pop | Q0 | Z |
| Q12 | + | Z | No push/pop | Q0 | Z |
| Q24 | + | Z | No push/pop | Q0 | Z |
| Q32 | + | z | No push/pop | Q0 | z |

SPECIAL CASE:

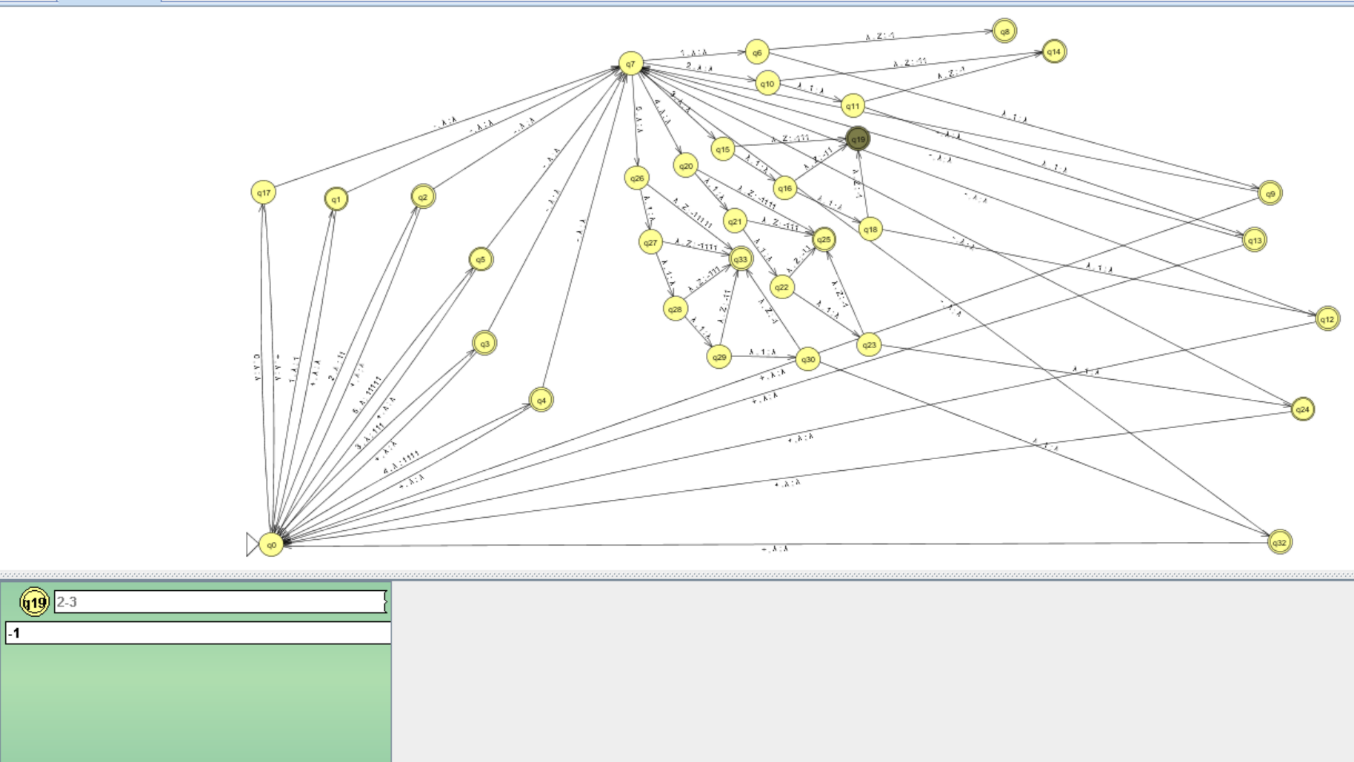
Our NPDA cannot move further and halts at the place where the result of the calculations is negative in between so to show the negative values we have included a special case where we can show the result in negatives when two numbers are subtracted but we should give any further operator or operand.

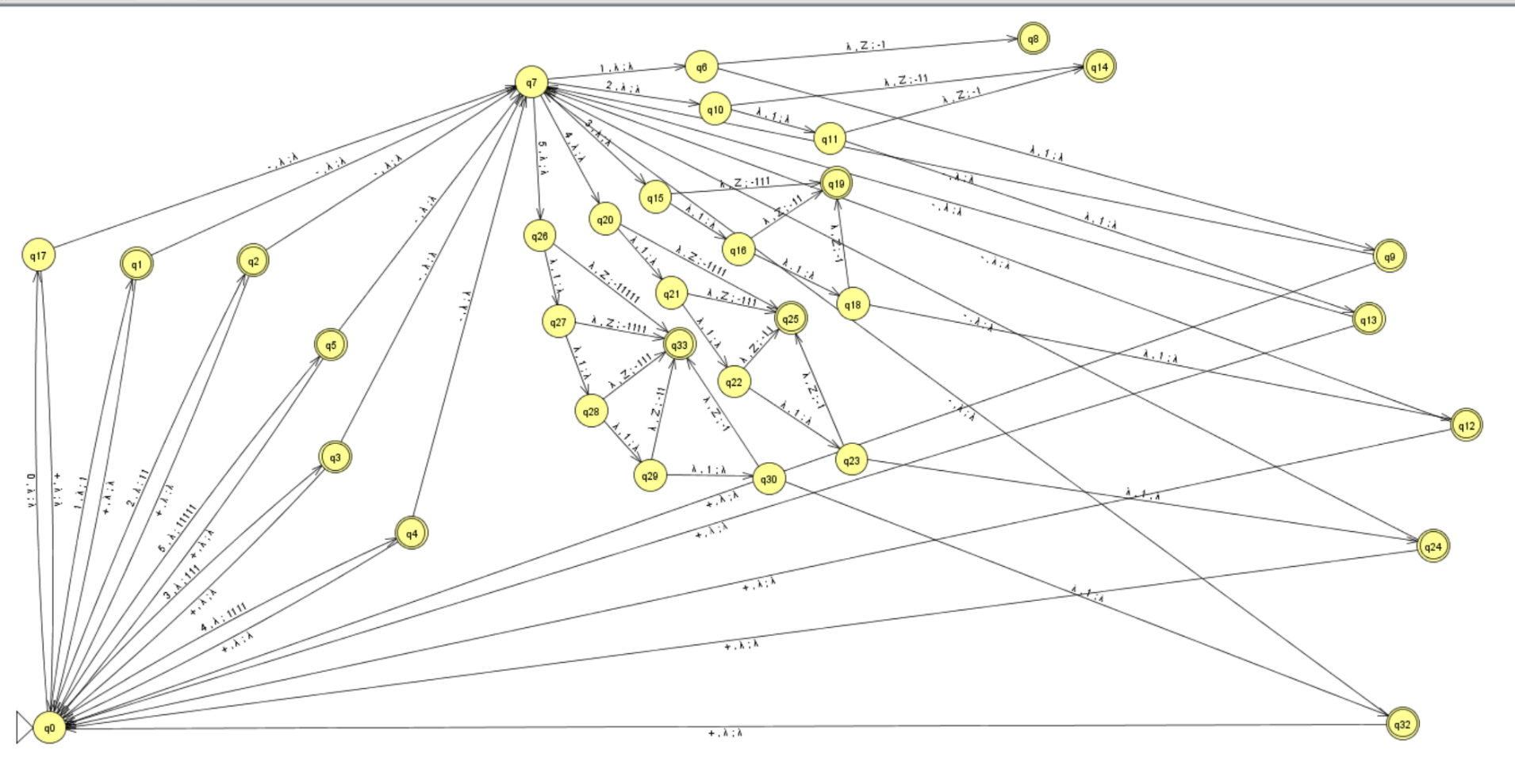
**INPUT:** 2-3

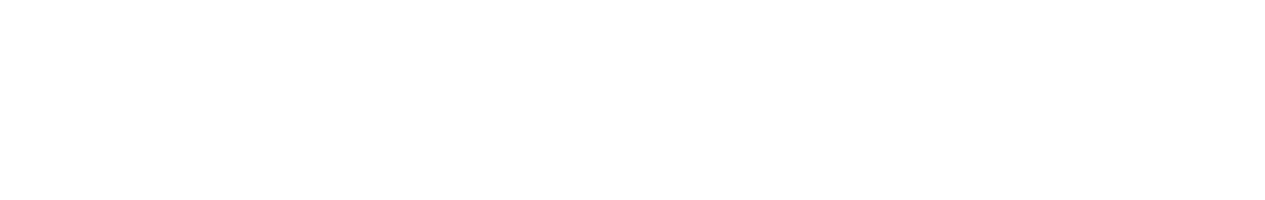
**OUTPUT:**



Final accepting state:







NPDA DESIGN

# CALCULATOR USING BINARY NUMBERS

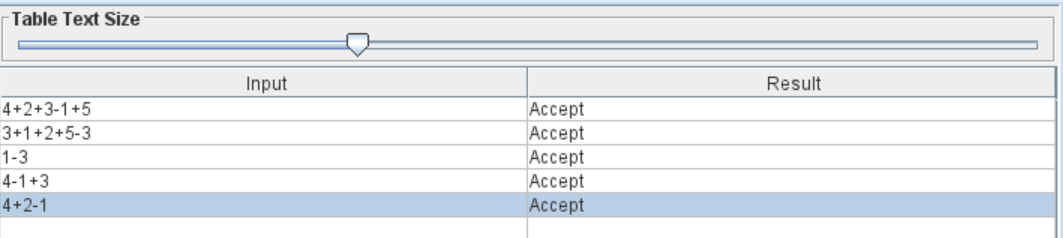
# 

# INPUTS:

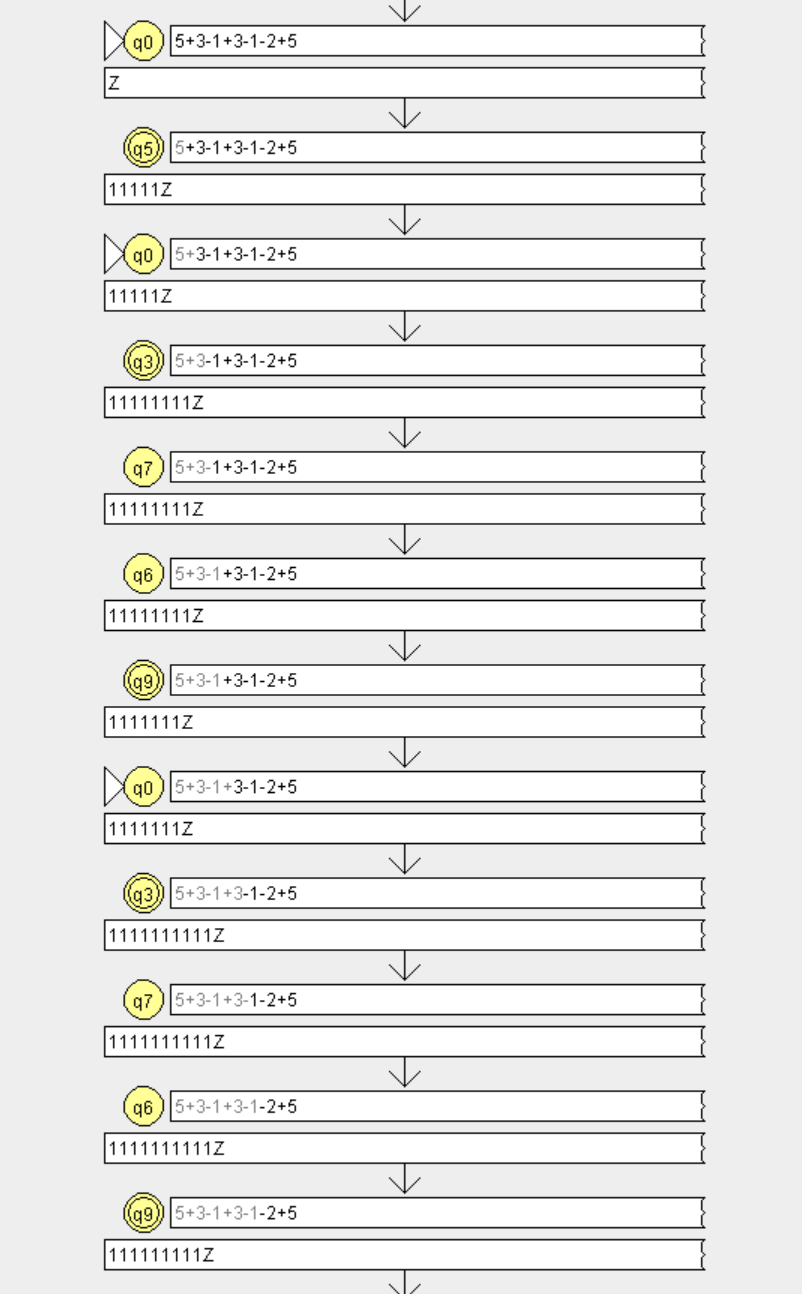
# 5+5-2+4-2-1

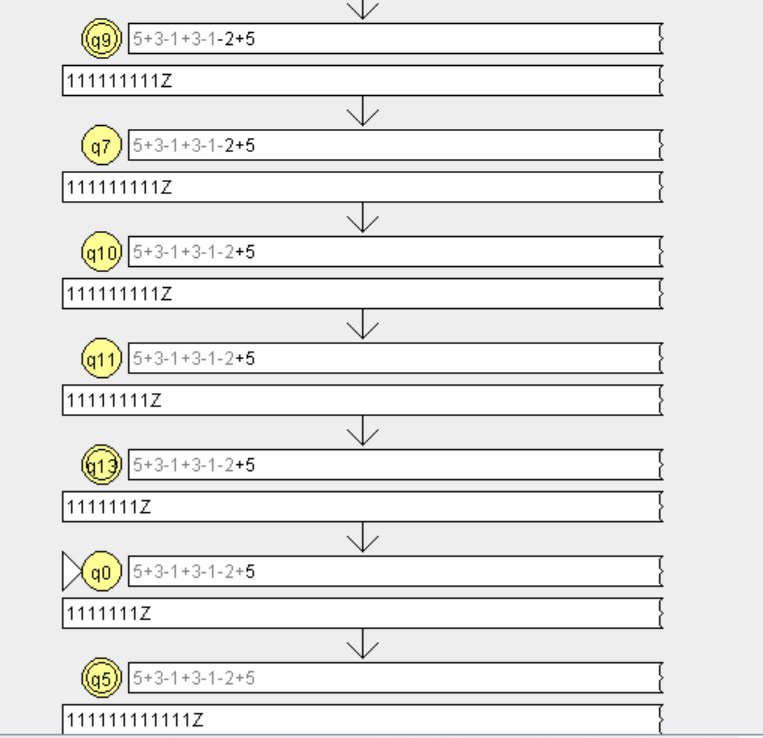
# 

Here are the few expressions which will be accepted in out NPDA:



**STACK CONTENT FOR THE INPUT: 5+3-1+3-1-1+5**

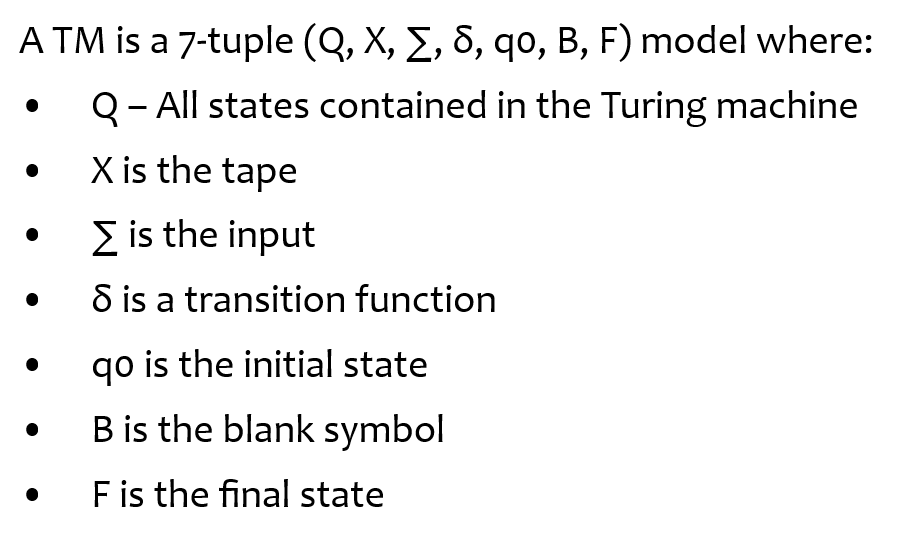


****

**TURING MACHINE**

A Turing Machine is a model consisting of a tape which has an indefinite length and is divided into several parts. These parts or cells consist of the input that are read.

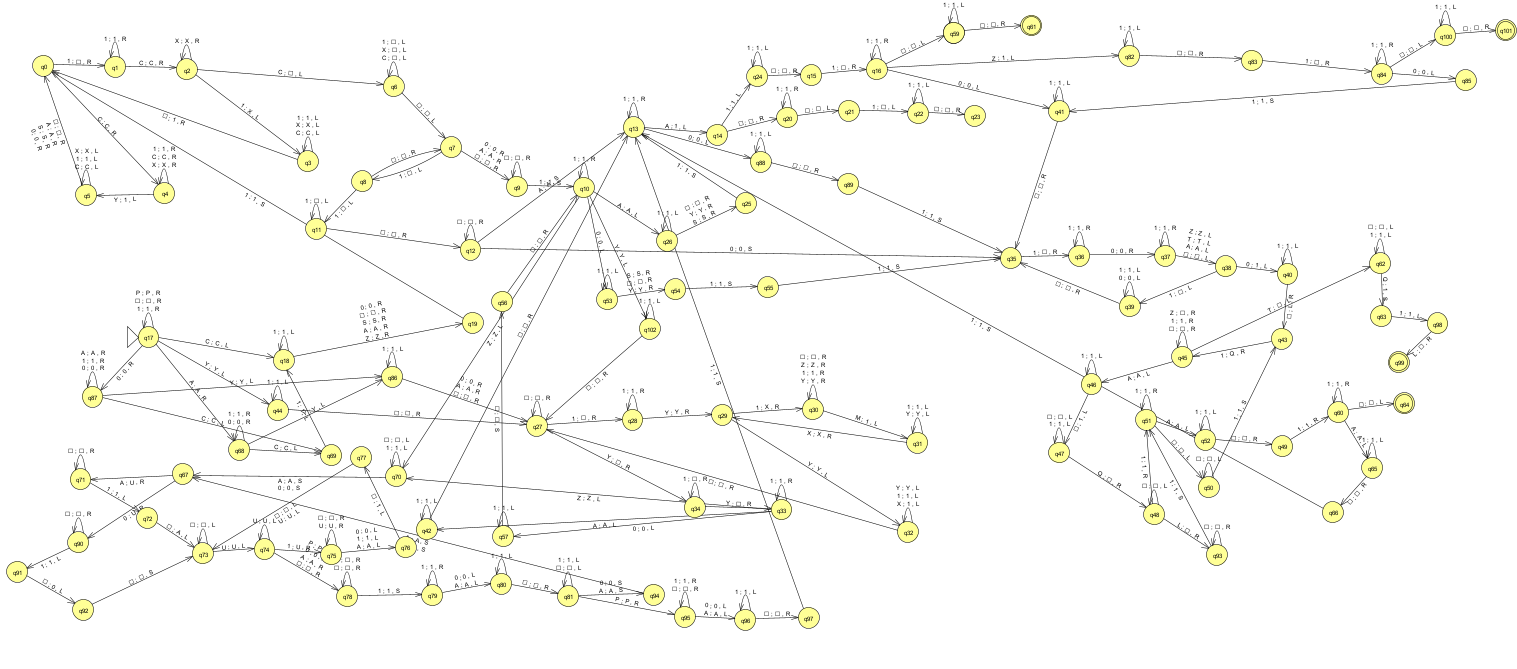
The state of a Turing Machine is stored in the state register. The format of the inputs given as input state, next state that replaces the input state and finally the right or left movement direction. If the final state is reached by this process, then the input is accepted, else it is rejected.

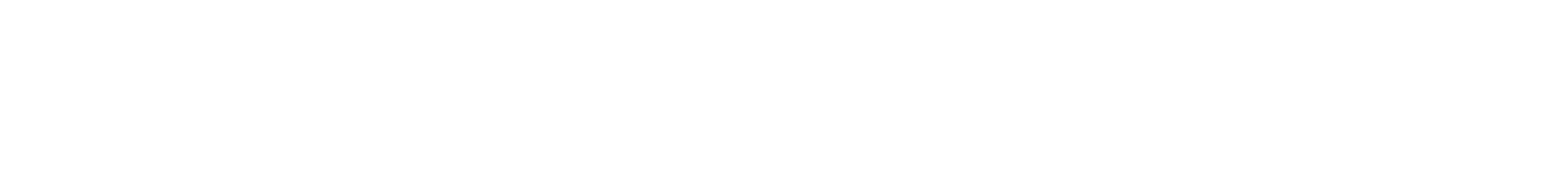


**Why Turing machine is better than NPDA/PDA**:

* Calculator is a device which generally gives an output. but a PDA/NPDA by default is a acceptor so a calculator is not possible with PDA/NPDA. Whereas Turing machine can also act as transducer so we can display the output and show to the user.
* PDA/NPDA can be used for expression evaluation. Whereas Turing machine can be a transducer or a acceptor so an Turing machine can be best used to implement a calculator
* So while applying bodmas rule, we have to compare operators for which we have to remember all the operators preceding it because we have to compare the precedence of the operator with the precedence of all the before operator in this case a stack cannot be used as it can just remember the precedence of the last occurring operator but not the entire thing.
* So, in this usage oof tape data structure is more useful so Turing machine is better than PDA.

Overall, Turing machines offer the required flexibility, symbol manipulation capabilities, and unbounded tape to model and implement the calculator





SINGLE TAPE TURING MACHINE

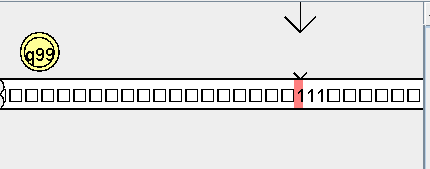
DESIGN

**FEW INPUTS AND OUTPUTS CASES :**

Possible Test Cases

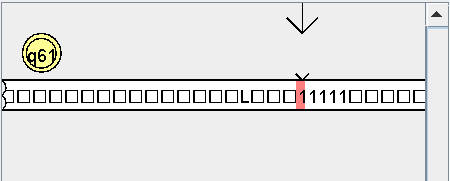
**Case 1:11C1111CYYA111011T**

**Expected :4/2+3-2=3**

****

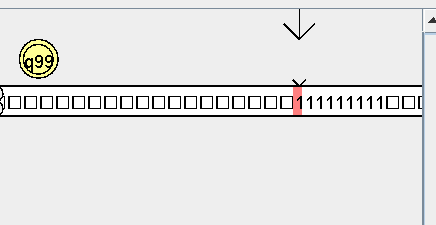
**Case 2:11C11111111CYYYY011A111**

**Expected :8/2-2+3=5**

****

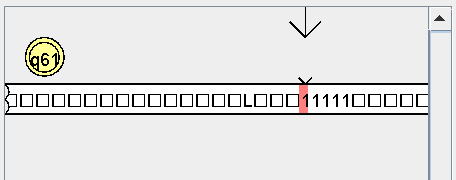
**Case 3:11Y1111YMMMMMMMMA111011T**

**Expected:2\*4+3-2=9**

****

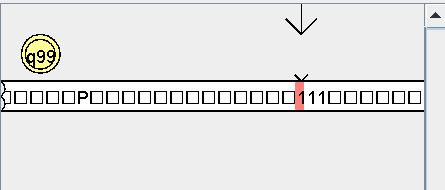
**Case 4:11Y11YMMMM011A111**

**Expected:2\*2-2+3=5**

****

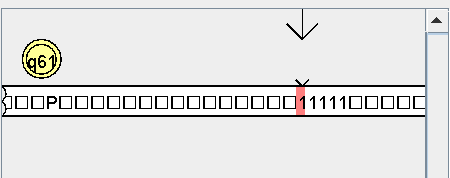
**Case 5:P111A11C1111CYYZ011T**

**Expected:3+4/2-2=3**

****

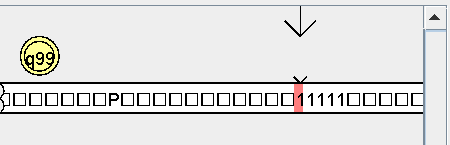
**Case 6:P1111011C1111CYYZA111**

**Expected:4-4/2+3=5**

****

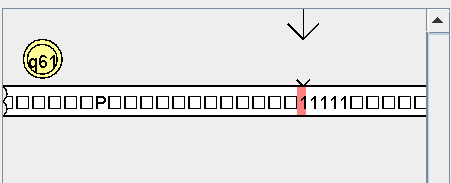
**Case 7:P111A11Y11YMMMMZ011T**

**Expected:3+2\*2-2=5**

****

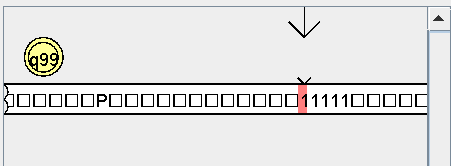
**Case 8:P1111011Y1YMMZA111**

**Expected:4-2\*1+3=5**

****

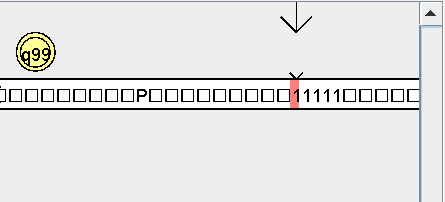
**Case 9:P111A1111011C1111CYYZT**

**Expected:3+4-4/2=5**

****

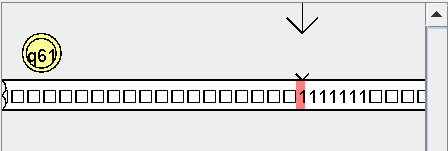
**Case 10:P111A1111011Y1YMMZT**

**Expected:3+4-2\*1=5**

****

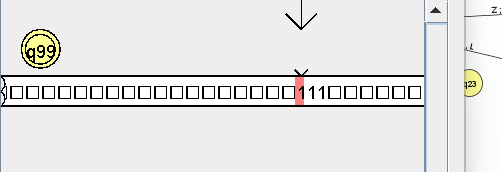
**Case 11:11C1111CYYY11YMMMMA111**

**Expected:4/2\*2+3=7**

****

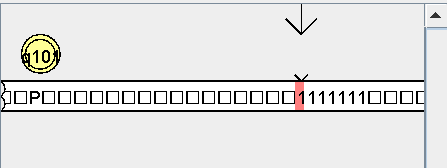
**Case 12:11C1111CYYY111YMMMMMM0111T**

**Expected:4/2\*3-3=3**

****

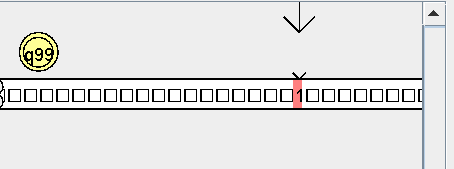
**Case 13:P111A11C1111CYYY11YMMMMZ**

**Expected:3+2\*4/2=7**

****

**Case 14:P11111011C1111CYYY11YMMMMZT**

**Expected:5-4/2\*2=1**



Challenges we faced are:

* To get a correct representation of a calculator by calculating for all types of inputs following bodmas considering complex mathematics topics like limits is next to impossible. Turing machine can provide a solution only for simple calculations which perform basic operations like addition, subtraction and multiplication, division and power.
* The solution of a Turing machine may not or will not provide an effective solution for a simple calculator, as considering a lot of assumptions we got a solution of more than 120 states. So a Turing machine is not that effective
* So if we perform a/b+c then here we need extra space for result if that extra space is not provided then the result of division will be stored in place of b+c . bits of result of division is overlapping with b+c which made us to make an assumption that memory should be taken as a input.
* In most of the case for example if we perform 3-2+1 then there are lot of blank spaces left between the result of subtraction and +1 due to which we needed extra states to move the result of subtraction beside +1 removing all the blanks between them
* In most of the cases the Turing machine goes into an infinite loop due to which we are forced to introduce P and T the start and end of an expression Many complications will be faced if we further add more conditions then Turing machine becomes ineffective and we need to look for further more solutions
* While developing a calculator using an NPDA diagram, we encountered several challenges. The first challenge arose is while evaluating expressions within the calculator. Specifically, if a subtractor operator appeared in the middle of the expression and the result turned out to be negative, we faced the dilemma of being unable to evaluate that particular expression further. This posed a significant hurdle in the calculator's functionality.
* Additionally, another challenge we faced was the limitation on using few operations. The calculator we designed could perform addition and subtraction operations. Multiplication and division operations were not supported, which restricted the range of mathematical calculations the calculator could handle.

THANK YOU