Student Declaration of Authorship

HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

Course Code and Name:	Foundations 2 (F29FB)
Type of Assessment:	Individual
Coursework Title:	Assignment
Student Name:	Chandrashekhar Ramaprasad
Student ID Number:	H00356126

Declaration of Authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission is expressed in my own words. Any uses made within this work of the ideas, writings, or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature: *Chandrashekhar Ramaprasad*

Date: *14/03/2023*

F29FB Assignment

Chandrashekhar Ramaprasad

H00356126 | cr2007

Green Number: 78

Contents

Section 1

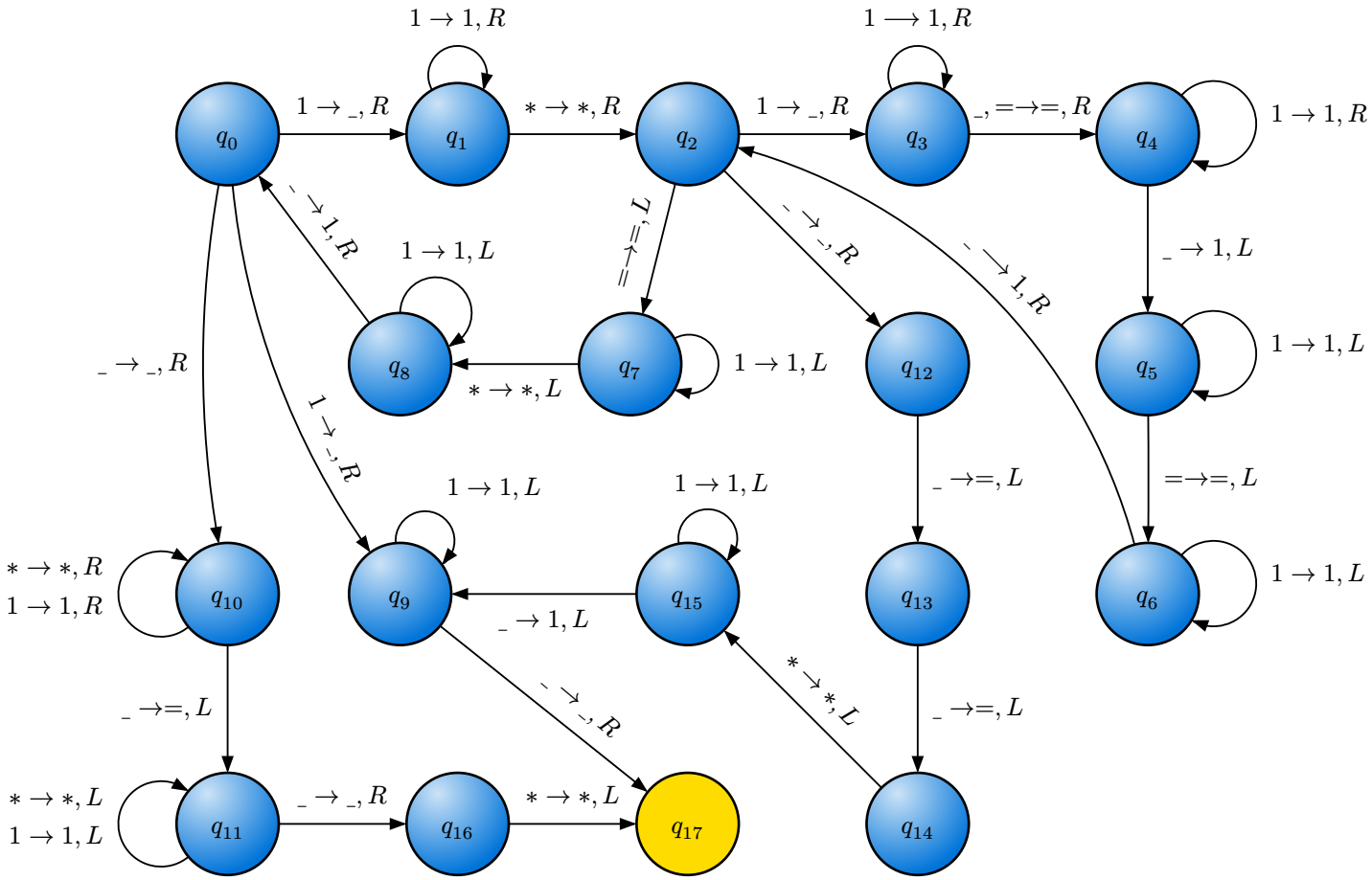


Figure 1: Turing Machine Graph

Section 2

The key difference between my proposed graph and the given graph is:

- The proposed graph has additional states q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}, q_{17} compared to the given graph
- The proposed TM graph has additional transitions in states q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}, q_{17} compared to the given graph
- The proposed graph writes the $*$ symbol in some of its transitions, while the given graph does not use that symbol

---

Section 3

States: $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8{\cdot}q_9{\cdot}q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}, q_{17}\}$
Where q_0 is the start state, and q_{17} is the halt state

Input Symbols: $\{1, *, ' '\}$ ($''$ is a blank symbol)

Tape Symbols: $\{1, *, ' ', =\}$, where $=$ is a special symbol

Start State: $q_0 \in Q$

Halt State: $q_{17} \in Q$

Actions Table:

Current State	Input Symbol	New Symbol	Move Direction	New State
q_0	1	$\wedge$	R	q_1
	*	*	L	q_9
	$\wedge$	$\wedge$	R	q_{10}
q_1	1	1	R	q_1
	*	*		q_2
q_2	1	$\wedge$	R	q_3
	$=$	$=$	L	q_7
	$\wedge$	$\wedge$	R	q_{12}
q_3	1	1	R	q_3
	$\wedge$	$=$		q_4
	$=$			
q_4	1	1	R	q_4
	$\wedge$		L	q_5
q_5	1	1	L	q_5
	$=$	$=$		q_6
q_6	1	1	L	q_6
	$\wedge$		R	q_2
q_7	1	1	L	q_7
	*	*		q_8
q_8	1	1	L	q_8
	$\wedge$		R	q_0
q_9	1	1	L	q_9
	$\wedge$	$\wedge$	R	q_{17}
q_{10}	1	1	R	q_{10}
	*	*		
	$\wedge$	$=$	L	q_{11}
q_{11}	1	1	L	q_{11}
	*	*		
	$\wedge$	$\wedge$	R	q_{16}
q_{12}	$\wedge$	$=$	L	q_{13}

Current State	Input Symbol	New Symbol	Move Direction	New State
q_{13}	$\wedge$	$\wedge$	L	q_{14}
q_{14}	$*$	$*$	L	q_{15}
q_{15}	$\wedge$	1	L	q_{15}
	1			q_{15}
q_{16}	$*$	$*$	L	q_{17}
q_{17}	Final (Accept) State			

Section 4

The Turing Machine graph was implemented in the Python programming language. Python was used to implement the above Turing Machine graph due to its simple syntax to represent the graph and easily manipulate the values provided. It is also very flexible and easy to define and manipulate complex data structures like Turing machines using the List data structure.

```python
def action(input_val: str, write: str, move: str, next_state: str):
    global tape_head, state
    if tape[tape_head] == input_val:
        tape[tape_head] = write
        state = next_state

        if move == 'L':
            tape_head -= 1
        else:
            tape_head += 1
        return True
    return False

input_text = "Make sure to enter your number as a set of ones\n(i.e. if the number if
2 * 3, enter it is '11*111')\nEnter the input:"
val: str = input(input_text)

length: int = len(val)
tape = ['^'] * (length*3)

i: int = 1
tape_head: int = 1

for n in range(length):
    if val[n] == '0' or val[n] == 'B':
        tape[i] = '^'
    else:
        tape[i] = val[n]
    i += 1

q0, q1, q2, q3, q4 = 'q0', 'q1', 'q2', 'q3', 'q4'
q5, q6, q7, q8, q9 = 'q5', 'q6', 'q7', 'q8', 'q9'

q10, q11, q12, q13, q14, q15, q16 = 'q10', 'q11', 'q12', 'q13', 'q14', 'q15', 'q16'
done: str = 'q17'

state: str = q0
r, l, b = 'R', 'L', '^'
accept: bool = False
old_tape_head = -1
tape_count = 0
```

Figure 2: Initialising variables and the function

```
while old_tape_head != tape_head:
    old_tape_head = tape_head
    print(tape, "Head: ", tape_head, "State: ", state)
    tape_count += 1

    if state == q0:
        if action('1', b, r, q1) or action(b, b, r, q10) or action('*', '*', l, q9):
            pass
    elif state == q1:
        if action('1', '1', r, q1) or action('*', '*', r, q2):
            pass
    elif state == q2:
        if action('1', b, r, q3) or action('=', '=', l, q7) or action(b, b, r, q12):
            pass
    elif state == q3:
        if action('1', '1', r, q3) or action(b, '=', r, q4) or action('=', '=', r, q4):
            pass
    elif state == q4:
        if action('1', '1', r, q4) or action(b, '1', l, q5):
            pass
    elif state == q5:
        if action('1', '1', l, q5) or action('=', '=', l, q6):
            pass
    elif state == q6:
        if action('1', '1', l, q6) or action(b, '1', r, q2):
            pass
    elif state == q7:
        if action('1', '1', l, q7) or action('*', '*', l, q8):
            pass
    elif state == q8:
        if action('1', '1', l, q8) or action(b, '1', r, q0):
            pass
    elif state == q9:
        if action('1', '1', l, q9) or action(b, b, r, done):
            pass
    elif state == q10:
        if action('1', '1', r, q10) or action('*', '*', r, q10) or action(b, '=', l, q11):
            pass
    elif state == q11:
        if action('1', '1', l, q11) or action('*', '*', l, q11) or action(b, b, r, q16):
            pass
    elif state == q12:
        if action(b, '=', l, q13):
            pass
    elif state == q13:
        if action(b, b, l, q14):
            pass
    elif state == q14:
        if action('*', '*', l, q15):
            pass
    elif state == q15:
        if action(b, '1', l, q9) or action('1', '1', l, q15):
            pass
    elif state == q16:
        if action('*', '*', l, done):
            pass
    else:
        accept = True
```

```python
if accept:
    print("\nSuccessfull!!")
    print("****************************************")
    print(f"Number of tapes used: {tape_count}")
    print("****************************************")
else:
    print(f"Input not accepted on state = {state}\n")
```

Section 5

This Turing Machine is designed to multiply a decimal number represented by a sequence of '1's with a separator in between the 2 inputs. It performs unary multiplication on these two numbers and adds the '=' sign with the resulting binary product. The machine begins in state q_0 and halts at the state q_{17} when the input is valid, and the multiplication operation is complete. The product of the input is stored in the same format, a sequence of 1s separated by a '$*$' symbol between the input and output followed by a '=' symbol with the product in the same format.

Section 6

The Turing machine is represented by a tape, a head that can read, write, and move across the tape, and a set of states and transition rules. The transition rules dictate the movement of the head and the symbols written on the tape depending on the current state and the symbol under the head.

<table>
<tr>
<td>The code initializes the tape with a "∧" symbol (blank) and asks the user to input two numbers separated by a "$*$" symbol.</td>
<td>

```python
input_text = "Make sure to enter your number as a set of ones\n(i.e. if the number if
2 * 3, enter it is '11*111')\nEnter the input:"
val: str = input(input_text)

length: int = len(val)
tape = ['^'] * (length*3)
```

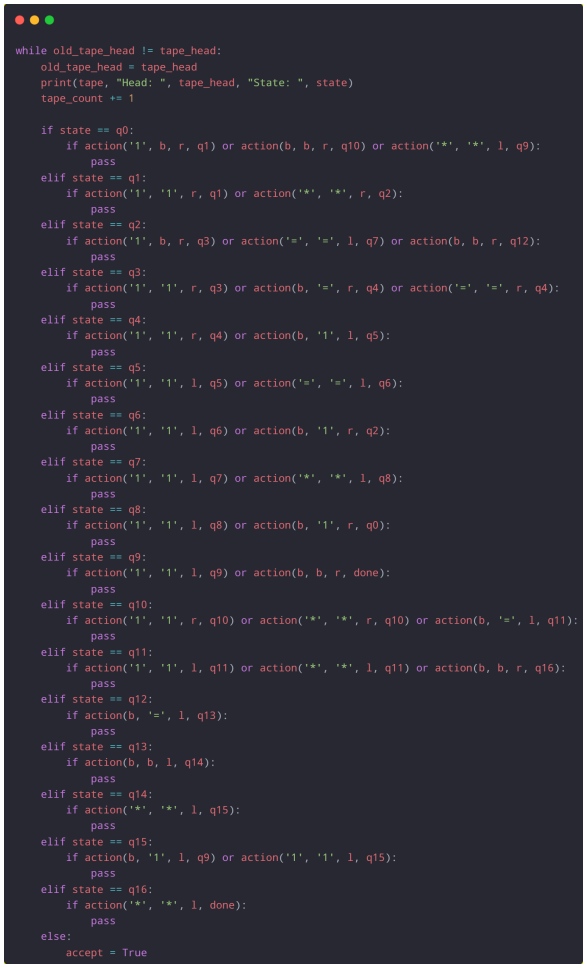
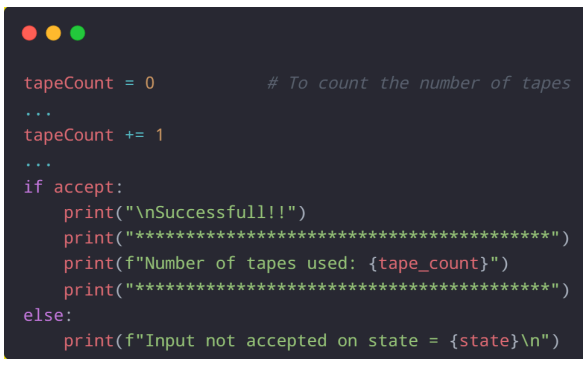
</td>
</tr>
<tr>
<td>The Turing machine uses a set of states and transition rules to perform the multiplication. The current state and symbol under the head determine which transition rule to apply.</td>
<td>

```python
while old_tape_head != tape_head:
    old_tape_head = tape_head
    print(tape, "Head: ", tape_head, "State: ", state)
    tape_count += 1

    if state == q0:
        if action('1', b, r, q1) or action(b, b, r, q10) or action('*', '*', l, q9):
            pass
    elif state == q1:
        if action('1', '1', r, q1) or action('*', '*', r, q2):
            pass
    elif state == q2:
        if action('1', b, r, q3) or action('=', '=', l, q7) or action(b, b, r, q12):
            pass
    elif state == q3:
```

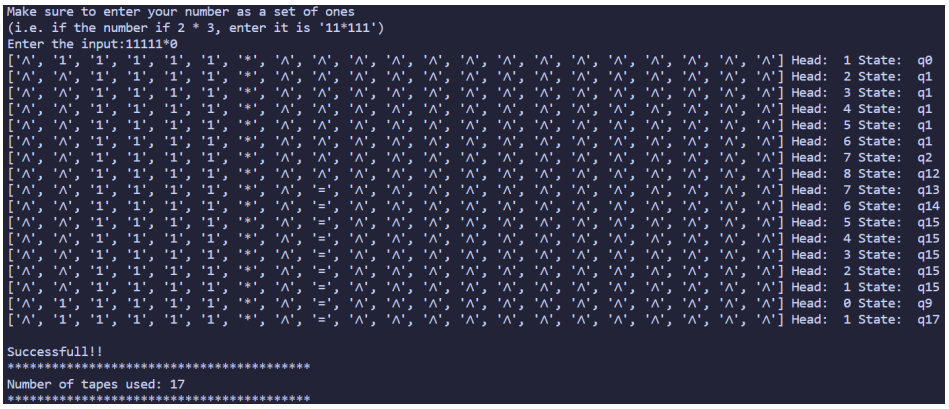
</td>
</tr>
<tr>
<td>The transition rules are represented by the "<code>action()</code>" function, which takes four arguments:
• the symbol under the head to be compared with the input value,
• the symbol to write on the tape,
• the direction to move the head (left or right),
• and the next state to transition to.
If the symbol under the head matches the input valie, the <code>action()</code> function writes the new</td>
<td>

```python
def action(input_val: str, write: str, move: str, next_state: str):
    global tape_head, state
    if tape[tape_head] == input_val:
        tape[tape_head] = write
        state = next_state

        if move == 'L':
            tape_head -= 1
        else:
            tape_head += 1
        return True
    return False
```

</td>
</tr>
</table>

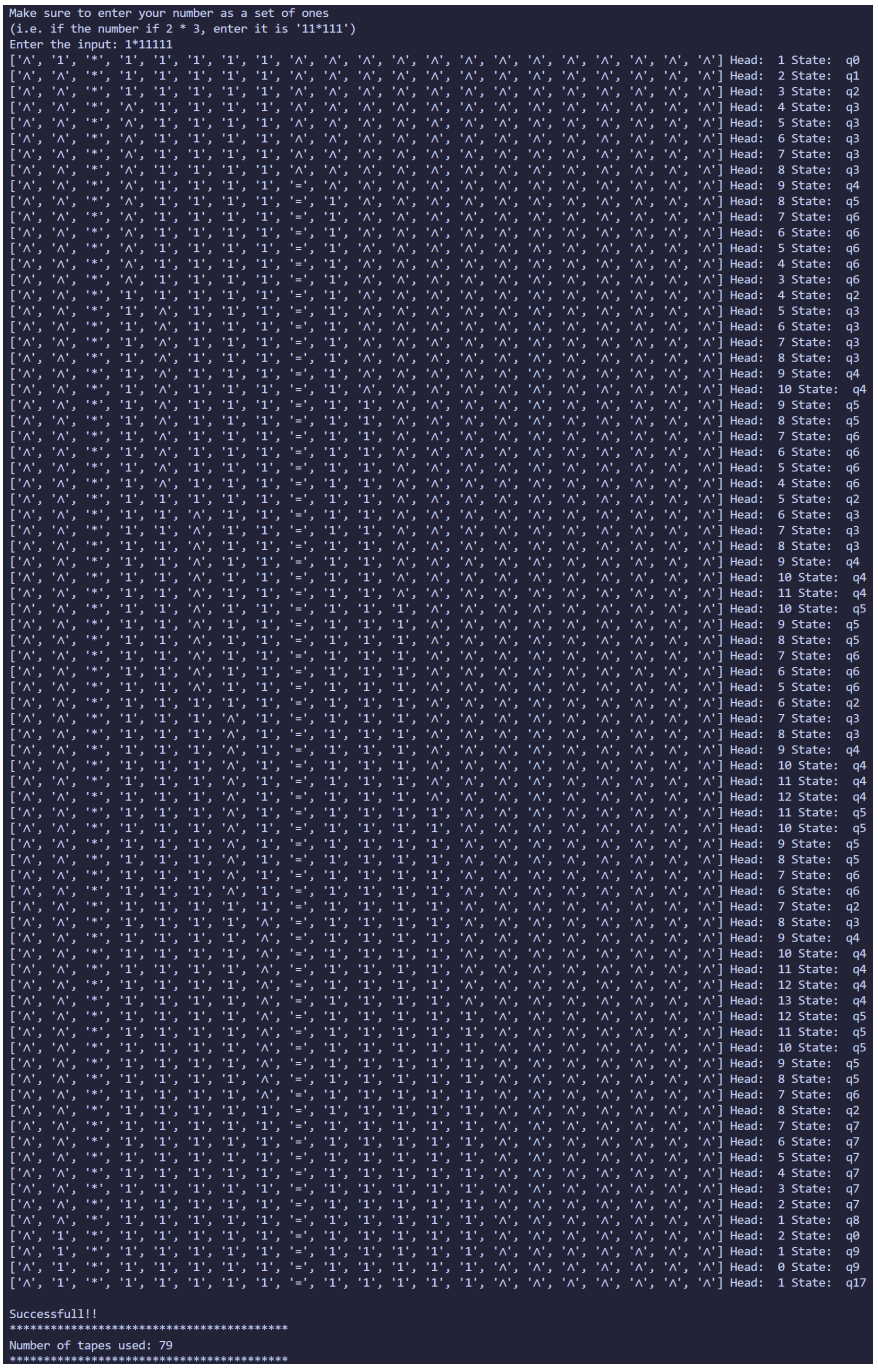
symbol on the tape, moves the head, and transitions to the next state. Otherwise it returns `False`	
The Turing machine transitions through several states, starting with "q_0" and ending with "q_{17}". Along the way, it performs various operations to multiply the two numbers on the tape. For example, the machine moves the head to the right until it encounters a "=" symbol, then moves it back to the left until it finds a "1" symbol, and then repeats this process until the multiplication is complete.	
The machine also keeps track of the number of tapes it has used and stops once the result is printed on the tape.	

Section 7

0×3 $0 \times 111)$	
5×0 (11111×0)	

2×3
(11×111)

```
@cr2007 ➜ /workspaces/F29FB-Assignment (main) $ python3 F29FB.py
Make sure to enter your number as a set of ones
(i.e. if the number if 2 * 3, enter it is '11*111')
Enter the input: 11*111
['^', '1', '1', '*', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  1 State:  q0
['^', '^', '1', '*', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  2 State:  q1
['^', '^', '1', '*', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  3 State:  q1
['^', '^', '1', '*', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  4 State:  q2
['^', '^', '1', '*', '^', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  5 State:  q3
['^', '^', '1', '*', '^', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q3
['^', '^', '1', '*', '^', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q3
['^', '^', '1', '*', '^', '1', '1', '=', '^', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  8 State:  q4
['^', '^', '1', '*', '^', '1', '1', '=', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q5
['^', '^', '1', '*', '^', '1', '1', '=', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q6
['^', '^', '1', '*', '^', '1', '1', '=', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  5 State:  q6
['^', '^', '1', '*', '^', '1', '1', '=', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  4 State:  q6
['^', '^', '1', '*', '1', '1', '1', '=', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  5 State:  q2
['^', '^', '1', '*', '1', '^', '1', '=', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q3
['^', '^', '1', '*', '1', '^', '1', '=', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q3
['^', '^', '1', '*', '1', '^', '1', '=', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  8 State:  q4
['^', '^', '1', '*', '1', '^', '1', '=', '1', '^', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  9 State:  q4
['^', '^', '1', '*', '1', '^', '1', '=', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  8 State:  q5
['^', '^', '1', '*', '1', '^', '1', '=', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q5
['^', '^', '1', '*', '1', '^', '1', '=', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q6
['^', '^', '1', '*', '1', '^', '1', '=', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  5 State:  q6
['^', '^', '1', '*', '1', '1', '1', '=', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q2
['^', '^', '1', '*', '1', '1', '^', '=', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q3
['^', '^', '1', '*', '1', '1', '^', '=', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  8 State:  q4
['^', '^', '1', '*', '1', '1', '^', '=', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  9 State:  q4
['^', '^', '1', '*', '1', '1', '^', '=', '1', '1', '^', '^', '^', '^', '^', '^', '^', '^'] Head:  10 State:  q4
['^', '^', '1', '*', '1', '1', '^', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  9 State:  q5
['^', '^', '1', '*', '1', '1', '^', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  8 State:  q5
['^', '^', '1', '*', '1', '1', '^', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q5
['^', '^', '1', '*', '1', '1', '^', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q6
['^', '^', '1', '*', '1', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q2
['^', '^', '1', '*', '1', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q7
['^', '^', '1', '*', '1', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  5 State:  q7
['^', '^', '1', '*', '1', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  4 State:  q7
['^', '^', '1', '*', '1', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  3 State:  q7
['^', '^', '1', '*', '1', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  2 State:  q8
['^', '^', '1', '*', '1', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  1 State:  q8
['^', '1', '1', '*', '1', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  2 State:  q0
['^', '1', '^', '*', '1', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  3 State:  q1
['^', '1', '^', '*', '1', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  4 State:  q2
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  5 State:  q3
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q3
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q3
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  8 State:  q4
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  9 State:  q4
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  10 State:  q4
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '^', '^', '^', '^', '^', '^', '^'] Head:  11 State:  q4
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  10 State:  q5
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  9 State:  q5
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  8 State:  q5
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q5
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q6
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  5 State:  q6
['^', '1', '^', '*', '^', '1', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  4 State:  q6
['^', '1', '^', '*', '1', '1', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  5 State:  q2
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q3
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q3
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  8 State:  q4
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  9 State:  q4
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  10 State:  q4
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  11 State:  q4
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  12 State:  q4
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  11 State:  q5
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  10 State:  q5
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  9 State:  q5
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  8 State:  q5
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q5
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q6
['^', '1', '^', '*', '1', '^', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  5 State:  q6
['^', '1', '^', '*', '1', '1', '1', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  6 State:  q2
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  7 State:  q3
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  8 State:  q4
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  9 State:  q4
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  10 State:  q4
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  11 State:  q4
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  12 State:  q4
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '^', '^', '^', '^', '^', '^'] Head:  13 State:  q4
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '1', '^', '^', '^', '^', '^'] Head:  12 State:  q5
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  11 State:  q5
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  10 State:  q5
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  9 State:  q5
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  8 State:  q5
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  7 State:  q5
['^', '1', '^', '*', '1', '1', '^', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  6 State:  q6
['^', '1', '^', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  7 State:  q2
['^', '1', '^', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  6 State:  q7
['^', '1', '^', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  5 State:  q7
['^', '1', '^', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  4 State:  q7
['^', '1', '^', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  3 State:  q7
['^', '1', '^', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  2 State:  q8
['^', '1', '1', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  3 State:  q0
['^', '1', '1', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  2 State:  q9
['^', '1', '1', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  1 State:  q9
['^', '1', '1', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  0 State:  q9
['^', '1', '1', '*', '1', '1', '1', '=', '1', '1', '1', '1', '1', '1', '^', '^', '^', '^'] Head:  1 State:  q17

Successfull!!
**************************************
Number of tapes used: 95
**************************************
```

1×5 (1×11111)	

Section 8

The program uses a while loop to iterate through the tape to perform computations until the tape head no longer moves. This is a reasonable approach since it ensures that the program terminates once no more computations can be performed. The program also uses a global variable to keep track of the state and the tape head position, which simplifies the code but could make it harder to maintain and modify.

Regarding the individual states, the program uses a combination of conditional statements and branching to determine the next state and tape head position. This approach is efficient since it reduces the number of iterations required to perform the computations.