Name: Harshit Jain
User ID: hmj5262

Problem 1

Algorithm 1 Checking for Infinite Strings in a DFA

```
Function HASINFINITESTRINGS(DFA):
visited_states ← empty set
stack ← [initial_state]
while stack is not empty do
  current_state ← pop(stack)
  Add current_state to visited_states
  for each transition (current_state, input_symbol, next_state) in DFA do
    if next_state is in visited_states then
      return true
    else if next_state is not in stack then
      push next_state to stack
    end if
  end for
end while
return false
```

1. **Function Definition**: The function definition for HASINFINITESTRINGS which takes a DFA as input and returns a boolean value indicating whether the DFA accepts an infinite number of strings.

2. **Initialization**: Inside the function, we initialize an empty set named *visited_states* to keep track of visited states during the DFS traversal. We also initialize a stack named *stack* with the initial state of the DFA.

3. **DFS Traversal**: The algorithm enters a while loop that continues until the stack is empty. In each iteration of the loop, it pops the top state (*current_state*) from the stack.

4. **Flag Visited States**: The current state is then added to the *visited_states* set to mark it as visited.

5. **Transition Exploration**: For each transition originating from the current state in the DFA, the algorithm checks whether the next state (*next_state*) has already been visited. If it has been visited, that means there is a loop, and the function returns **true** indicating that the DFA accepts an infinite number of strings. If the next state has not been visited, it is pushed onto the stack for further exploration.

6. **Termination**: If the while loop completes without finding any loop, the function returns **false**, indicating that the DFA accepts a finite number of strings.