CS868: Compilers
Classwork – Finite State Machines

1. **QUESTION LABEL:** *FSM 1*

   Implement a finite state machine that accepts the word 'in'. For example:

```
recognise("in") → true
recognise("out") → false
recognise("int") → false
```

2. **QUESTION LABEL:** *FSM 2*

   Implement a FSA that accepts the word 'in' even if it is a prefix of the provided string. It returns the resultant index.

```
recognise("in") → (true, 2)
recognise("out") → (false, _)
recognise("int") → (true, 2)
```

3. **QUESTION LABEL:** *FSM 3*

   Implement an FSA that accepts the word 'in' based on the the index provided. For example:

```
recognise("fin", 1) → (true, 3).
```

4. **QUESTION LABEL:** *FSM 4*

   Implement an FSA that accepts the pattern 'int+' based the index provided. For example:

```
recognise("fintttto", 1) → (true, 7)
```

5. **QUESTION LABEL:** *FSM 5*

   Implement a recogniser that uses the above two FSAs to identify 'in' and 'int+'. If 'in' is recognised (IN, p) is returned. If 'int+' is recognised then (INTP, p) is returned. Here, p is the position of the first unconsumed character. Please note that if both the FSAs succeed, then the one which has consumed a larger portion of the input is considered to win. For example:

```
recognise("finp", 1) → (IN, 3).
recognise("fintttto", 1) → (INTP, 7)
```

6. **QUESTION LABEL:** *FSM 6*

Implement a recogniser that uses the above two FSAs to identify 'in' and 'into'. If 'in' is recognised (IN, p) is returned. If 'into' is recognised then (INTO, p) is returned. Here, p is the position of the first unconsumed character. Please note that if both the FSAs succeed, then the one which has consumed a larger portion of the input is considered to win. For example:

```
recognise("finp", 1) → (IN, 3).
recognise("finto", 1) → (INTP, 5)
recognise("fintp", 1) → (IN, 3)
```