Question 1: Using a Two-Dimensional Tape

Imagine a modified Post-Turing language, called 2D Post-Turing Language, which works on a two-dimensional tape, which extends infinitely in all directions. To do computations effectively, this language has two additional instructions: UP and DOWN.

Write a 2D Post-Turing program using the alphabet \( A = \{F\} \), so the only symbols it works with are \( F \) and the usual blank \( B \). The only thing your program does is to fill the entire two-dimensional tape with the symbol \( F \).

Since the tape is infinite, the best method for filling each and every square of it with the symbol \( F \) is to proceed on a spiral path:

```
<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
```

Here, the tapehead starts in the center and then replaces all Bs with Fs on a spiral path. This way, it will eventually reach any square on the tape and write an \( F \) into it. Since the tape is infinite, the program never terminates.

You can use macros if you provide the code defining them.
Question 2: The Turing Machine Competition!

Build a Turing machine on the alphabet $A = \{a, b\}$ that computes a function $f(x)$ strictly. $f(x)$ sorts the symbols in the input string in the order $a, b$. For example,

- $f(bba) = abb$
- $f(bbab) = abbb$
- $f(aabababb) = aaaaabbbb$
- $f(0) = 0$

Hint: The bubble sort algorithm may be the easiest one to implement as a Turing machine.

Write down the Turing machine in quadruple notation and as a state transition diagram. Also give the sequence of configurations during the computation of $f(aba)$.

Whoever builds the Turing machine with the fewest internal states that correctly computes $f$ will get some bonus points!