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:

```
B B B B B B B
B B → → → ↓ B
B B ↑ → ↓ ↓ B
B B ↑ ↑ ↓ ↓ B
B B ↑ ← ← ↓ B
B B B ... ← ← B
B B B B B B B
```

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.
UP
[U1]  DOWN
[U2]  PRINT F
     UP
     RIGHT
     IF B THEN R1
     LEFT
     GOTO U2
[R1]  LEFT
[R2]  PRINT F
     RIGHT
     DOWN
     IF B THEN D1
     UP
     GOTO R2
[D1]  UP
[D2]  PRINT F
     DOWN
     LEFT
     IF B THEN L1
     RIGHT
     GOTO D2
[L1]  RIGHT
[L2]  PRINT F
     LEFT
     UP
     IF B THEN U1
     DOWN
     GOTO L2
Question 2: The Turing Machine Competition!

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

Quadruples:

$q_1 \quad B \quad R \quad q_2$
$q_1 \quad a \quad R \quad q_2$
$q_1 \quad b \quad R \quad q_3$
$q_2 \quad B \quad L \quad q_4$
$q_2 \quad a \quad R \quad q_2$
$q_2 \quad b \quad R \quad q_3$
$q_3 \quad B \quad L \quad q_4$
$q_3 \quad a \quad b \quad q_5$
$q_3 \quad b \quad R \quad q_3$
$q_4 \quad a \quad L \quad q_4$
$q_4 \quad b \quad L \quad q_4$
$q_5 \quad b \quad L \quad q_6$
$q_6 \quad b \quad a \quad q_7$
$q_7 \quad a \quad L \quad q_1$
Haskell:

sortMachine :: Machine
sortMachine = [(1, 'B', R, 2), (1, 'a', R, 2), (1, 'b', R, 3), (2, 'B', L, 4), (2, 'a', R, 2),
(4, 'b', L, 4), (5, 'b', L, 6), (6, 'b', P 'a', 7), (7, 'a', L, 1)]
Sequence of configurations:

Baba
  \[ \uparrow \]
  q_1

aba
  \[ \uparrow \]
  q_2

aba
  \[ \uparrow \]
  q_2

aba
  \[ \uparrow \]
  q_2

aba
  \[ \uparrow \]
  q_3

abb
  \[ \uparrow \]
  q_5

abb
  \[ \uparrow \]
  q_6

aab
  \[ \uparrow \]
  q_7

Machine halts. Output is aab.