Second Bonus Homework for Languages & Machines

Instructions. The solutions to these exercises must be handed in by Thursday, April 2, 17:30, via Canvas (a scan of handwritten solutions is ok). You should hand in your solutions in pairs of two students, with the student group Bonus Homework L&M. Clearly mark your name(s) and student number(s) on the top of the sheet.

Exercises. Consider the language

$$L = \{1^n \mid \exists\, k \in \mathbb{N} \colon n = 2^k\},$$

i.e. the language of the powers of two in unary representation. This language contains, for example, the words 1, 11 and 11111111, but neither 111 nor 11111111111.

1. Prove that L is not context-free.
2. Draw a standard Turing machine with a two-way tape that decides L, using only $\Gamma = \{0, 1, B\}$ as tape alphabet and at most $|Q| = 8$ states. Briefly explain how your machine works.

   For every state or group of related states, your explanation should say in a few words (e.g. "move back to start of tape" or "replace every blank by 1*") what its purpose is in deciding* L*. You may annotate your machine drawing with these explanations (in a clean and understandable way), or write a brief text below your machine drawing.*

For the experts (this is not part of the homework, do not hand in a solution):
What is the complexity (O or Θ) of the algorithm to recognise powers of two in *unary* representation that you implemented with your Turing machine, i.e. roughly how many transitions does a computation of your machine take for input number n (input word 1^n) in the worst case? Can you construct a Turing machine of strictly lower complexity that decides whether an input number n given in *binary* representation (with tape alphabet $\{0, 1, B\}$) is a power of two? What is its complexity in terms of n, and what is it in terms of the length of the (binary) input word?