PART II

- A button is broken...
- If you want to get a letter on that broken button:
    - Hold down # pound key
    - Click up arrow key to make menu option appear
    - A menu option will appear on screen displaying from the first letter on the broken button to the last letter on the broken button
    - Clicking the up and down arrow keys will cycle through the letters

Added tests:

Rule 8: If the letter is one from the broken button, and it is the first letter of the current tested word, it takes **0s** to press the first button # (pound key) and takes **0.25s** to press the up arrow key. Therefore it takes a total of **0.25s**.

Rule 9: When attempting to type a letter from the broken button, the wait time from any button to the # (pound key) is **0.25s**. Even if the previous button was the # (pound key), it does not take any added time to press the button again since it must only be released and re-pressed. This is a clone of project rule 2, however for broken letters, Rule 4 does not apply (the **0.5s** wait time).

Hand-written tests:
Broken digit: 2 (abc)

adgjm: 1.25s
a: 0.25s (Rule 8, takes 0s to press # + pressing up arrow)
ad: 0.25s + 0.25s (Rule 2) = 0.5s
adg: 0.5s + 0.25s (Rule 2) = 0.75s
adgj: 0.75s + 0.25s (Rule 2) = 1s
adhjm: 1s + 0.25s (Rule 2) = 1.25s

aaaaa: 2.25s
a: 0.25s (Rule 8)
aa: 0.25s + 0.25s (Rule 9) + 0.25s (Rule 8) = 0.75s
aaa: 0.75s + 0.25s (Rule 9) + 0.25s (Rule 8) = 1.25s
aaaa: 1.25s + 0.25s (Rule 9) + 0.25s (Rule 8) = 1.75s
aaaaa: 1.75s + 0.25s (Rule 9) + 0.25s (Rule 8) = 2.25s

aLErt: 6.5s
a: 0.25s (Rule 8)
aL: 0.25s + 0.25s (Rule 2) + 2 * 0.25s (Rule 3 applied twice) + 2s (Rule 2) = 3s
aLE: 3s + 0.25s (Rule 2) + 1 * 0.25s (Rule 3) + 2s (Rule 2) = 5.5s
aLEr: 5.5s + 0.25s (Rule 2) + 2 * 0.25s (Rule 3 applied twice) = 6.25s
aLEt: 6.25s + 0.25s (Rule 2) = 6.5s

wtpmj: 1s
w: 0s (Rule 1, first letter takes 0s)
wt: 0s + 0.25s (Rule 2) = 0.25s
wtp: 0.25s + 0.25 (Rule 2) = 0.5s
wtpm: 0.5s + 0.25 (Rule 2) = 0.75s
wtpmj: 0.75s + 0.25 (Rule 2) = 1s

Therefore wtpmj is the shortest time of these hand calculations and this aligns with the program we wrote.

PART III

In order to improve the overall use of the phone keypad, our group decided to consider the amount of buttons available (12), as well as the total functionality needed to be maintained. To conserve all functionality on the dial pad and simultaneously increase typing speed, we rearranged the special characters and letters, shifting values from some buttons to others. In the provided diagram, the newly proposed design for the keypad implementation is shown. Our group decided to place the space character, star key, pound key, and voicemail function on the “1” key. This creates a simple cycling function key at the top of the dial pad - an elegant and compact solution to the pad’s special uses. These functions were ordered on the key cycle from most to least frequently used, to further increase efficiency. Moreover, the letters are spread out over the remaining keys. With more keys dedicated to letters, each can have less characters, and therefore a shorter cycle. We researched commonly used letters, and thought that we could organize the vowels to be the first letters on their respective keys. This would prioritize common letters and phrases, and allow them to be typed with maximum efficiency and speed. Using this technique, the maximum number of letters on a single key is 3, whereas the original design had 4. Overall, our group’s new design for the phone’s dial pad takes into account a user’s experience and ease of use, while also lowering typing time and maintaining all core functionality.

Figure 1: Improved design for phone dial pad (Group P-11)