Algorithm: Proxy-Tuning of Language Model
/* Applies proxy-tuning to adjust logits of a target model in an NLP task */

Input: input_ids, large_model_base M, small_model_tuned M^+, small_model_untuned M^-,
Output: generated_text, a sequence of tokens

Hyperparameters: max_length (maximum generation length), n (number of tokens to generate)

Parameters: Θ includes all parameters for the large_model_base M, small_model_tuned M^+, and small_model_untuned M^-.

1. Initialize generated_tokens as an empty list
2. Encode input_text into input_ids using tokenizer Θ_tokenizer

// Perform token-wise proxy-tuning and text generation
3. For t in [1, ..., n]:
    a. Obtain large_model_base, small_model_tuned, and small_model_untuned logits:
        i. large_base_logits ← large_model_base M (input_ids). logits with parameters Θ_large_model_base
        ii. small_tuned_logits ← small_model_tuned M^+ (input_ids). logits with parameters Θ_small_model_tuned
        iii. small_untuned_logits ← small_model_untuned M^- (input_ids). logits with parameters Θ_small_model_untuned

    b. Proxy-tuning adjustment:
        i. Δlogit_offsets ← small_tuned_logits - small_untuned_logits
        ii. logits' ← large_base_logits + Δlogit_offsets

    c. Normalize the logits for next token prediction:
        i. predictions ← softmax (logits', axis=-1)

    d. Select the next token:
        i. next_token_id ← argmax(predictions)
        ii. Append next_token_id to generated_tokens

    e. Update input_ids with next_token_id for the next iteration

4. Decode the sequence of generated_tokens into text using Θ_tokenizer

5. **Return** generated_text