- O objetivo do trabalho é implementar a função `justifica`:

  ```
  justifica :: String -> String
  ```

  que recebe como entrada uma string contendo um texto em várias linhas e devolve o mesmo texto justificado pela maior linha. Exemplo de texto:

  ```
  texto = "RUBIÃO fitava a enseada -- eram oito horas da manhã.\n
  Quem o visse com os polegares metidos no cordão do chambre à janela de uma\n
  grande casa de Botafogo cuidaria que ele admirava aquele pedaço de água\n
  quieta mas em verdade vos digo que pensava em outra coisa.\n
  Cotejava o passado com o presente. Que era há um ano?\n
  Professor. Que é agora? Capitalista! Olha para si para as chinelas\n
  (umas chinelas de Túnis que lhe deu recente amigo Cristiano Palha) para a casa\n
  para o jardim para a enseada para os morros e para o
  céu e tudo desde as chinelas\n
  até o céu tudo entra na
  mesma sensação de propriedade."
  ```

- Exemplo de uso:

  ```
  Prelude> putStr (justifica texto)
  "RUBIÃO fitava a enseada -- eram oito horas da manhã.
  Quem o visse com os polegares metidos no cordão do chambre à janela de uma
  grande casa de Botafogo cuidaria que ele admirava aquele pedaço de água
  quieta mas em verdade vos digo que pensava em outra coisa.
  Cotejava o passado com o presente. Que era há um ano?
  Professor. Que é agora? Capitalista! Olha para si para as chinelas
  (umas chinelas de Túnis que lhe deu recente amigo Cristiano Palha) para a casa
  para o jardim para a enseada para os morros e para o céu e tudo desde as chinelas
  até o céu tudo entra na mesma sensação de propriedade."
  ```

- Dicas:
  (1) Quebrar o problema em problemas menores:

  ```
  separaLinhas :: String -> [String]
  tamanhoMaiorLinha :: [String]-> Int
  separaPalavras :: String -> [String]
  insereEspacos :: Int -> String -> String
  (...)
  ```

  (2) As funções `takeWhile` e `dropWhile` podem ser usadas para quebrar as strings em strings menores