```
import string
import numpy as np

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Activation
import keras


# TODO: fill out the function below that transforms the input series
# and window-size into a set of input/output pairs for use with our RNN model
def window_transform_series(series, window_size):
    # containers for input/output pairs
    X = []
    y = []
    for i in range(len(series) - window_size):
        X.append(series[i:(i + window_size)])
        y.append(series[(i + window_size)])

    # reshape each
    X = np.asarray(X)
    X.shape = (np.shape(X)[0:2])
    y = np.asarray(y)
    y.shape = (len(y), 1)

    return X, y


# TODO: build an RNN to perform regression on our time series input/output data
def build_part1_RNN(window_size):
    model = Sequential()
    model.add(LSTM(5, input_shape=(window_size, 1)))
    model.add(Dense(1))

    return model


# TODO: return the text input with only ascii lowercase and the punctuation
# given below included.
def cleaned_text(text):
    punctuation = ['!', ',', '.', ':', ';', '?', ' ']
    allowed = string.ascii_lowercase + ''.join(punctuation)
    new_text = []
    for c in text:
        if c not in allowed:
            c = ' '
        new_text.append(c)
    return ''.join(new_text)


# TODO: fill out the function below that transforms the input text and
# window-size into a set of input/output pairs for use with our RNN model
def window_transform_text(text, window_size, step_size):
    # containers for input/output pairs
    inputs = []
    outputs = []

    for i in range(0, len(text) - window_size, step_size):
        inputs.append(text[i:i + window_size])
        outputs.append(text[i + window_size])

    return inputs, outputs


# TODO build the required RNN model:
# a single LSTM hidden layer with softmax activation,
# categorical_crossentropy loss
def build_part2_RNN(window_size, num_chars):
    model = Sequential()
    model.add(LSTM(200, input_shape=(window_size, num_chars)))
    model.add(Dense(num_chars))
    model.add(Activation('softmax'))

    return model
```