For assignment 5 coding part, I used the following paper:
https://archiv.infsec.ethz.ch/education/fs08/secsem/bleichenbacher98.pdf

This is a better way than brute force because of the breaking up in intervals and merging the overlapped ones.
Public key was given to us as (e,N) in pub_key of rsa file where modulus_size was 96. The ideas are from paper with the assumption of the existence of an oracle which can perform check_padding (callable from rsa), and hence just by using a function which gives information about “some error” (here format) we could recover message.
The message is broken into chunks with assumption it cannot exceed modulus_size - 11.
The padding function was given to us and in general case it is pseudo random. Padding size is modulus_size - 3 - message size and first to blocks of overall ct are spl blocks. Padding length is also min 8 with non 0 blocks. So finally when we find message, we skip first 10 and look for a block with value 0 so that subsequent blocks give message.
Since we were given a conforming message we don’t need blinding described in the step 1 of paper. So we start by making our own ceil , floor and power (as used in python but modular) and set initial s as ceil(n/3*byteLen) and initial interval_list as (2*byteLen, 3*byteLen-1). Then since there is only 1 interval, we calculate the min s greater than initial s such that c*(s^e) mod N is also pkcs conforming.
Whenever we find a new s, we use step 3 of updating intervals (exadct formula is mentioned). Now number of interval in interval_list can change from 1 to more than 1 or same. So if more than 1 interval is there, we find corresponding to prev value of s, which is the new s greater than prev s and satisfies c*(s^e) mod N is also pkcs conforming(). If only 1 interval left, then we check is the interval containing more than 1 value or only 1 value. If it has only 1 value, by correctness of reducing our search space and keeping message in the interval, we can claim this message is exactly the element of interval (which is now singleton).
Otherwise, if this interval has more elements, perform step 2c with exact formula mentioned in the paper.
This is inside a while loop so this keeps on happening and after 2b or 2c, update_interval (step3) is called so finally the code terminates with the message given as a list of int. This takes significant time as significant computing power is used.
Once I get the message, I strip of the first 10 blocks and look for the first 0 block post which I get the message.
This message can be then converted to string easily (a commented code at bottom)
The various functions are built according to various steps at each stage. In update interval, one extra for loop is present to check for overlapping intervals to improve efficiency (as compared to naive implementation which just appends without checking for overlapping interval)