DNSSEC IMPLEMENTATION (PART B of HW1)

The code begins by defining a list of IP addresses of root servers, which are the 13 geographically distributed Root DNS Servers.
A set containing the key,value pairs of DS Record's RRSet's Digest types and their respective Hashing Algorithms is declared.
The user gives in 2 inputs : The Domain Name and the type of Query they want to input. For this exercise, we only support Query of rdtype "A"
The first and the most important function that is called is:
Main_dig() :
This is the central logic of the code which takes in the Domain type and the query type as the input from the user.
It first begins by resolving the root "." zone
Two disparate queries are formed, one for obtaining the current server's DNSKey response, and one for obtaining its DS Record response. These queries are then fed into **GetResponse()** function, which is an iterative query resolver that basically returns the dns.query.udp's response pertaining to the query that was sent to it by the main_dig() function.
After obtaining the respective response bodies, the ObtainValues() function is called.
ObtainValues() : this function's job is to return these 5 values for the upcoming DNSSEC verification :

a. DNSKey's RRSet (it contains PubZSK and PubKSK for that particular zone)
b. DNSKey's RRSig
c. Ksk (extracted from DNSKey's RRSet)
d. DSRecord's RRSet/ Answer (it contains hashed PubKSK for the child/next server/zone)
e. DSRecord's RRSig

Once these 5 values are returned into the Main_dig() function, the actual DNSSEC Verification will take place.
The following three conditions should be satisfied so as to validate that particular server/zone:

1. The **DNSKey RRSet of the current server should be validated by decrypting its RRSig** with the current server's PubKSK: we use dns.dnssec.validate by passing the DNSKey RRSet and the DNSKey RRSig for this

2. The **DSRecord/A RRSet of the current server should be validated by decrypting its RRSig** with the current server's PubZSK we use

dns.dnssec.validate by passing the DSRecord RRSet and the DSRecord RRSig for this

3. The current server zone is verified by **matching the hashes**: we perform hashing on the current server's PubKSK based on the hashing algorithm specified in the DS Record's RRSet's Digest type, and compare it with the value provided by the parent server's DSRecord RRSet. In case of the root server, as we do not have any parent referring to it, we use hashed record for the root server (reference https://github.com/iana-org/get-trust-anchor)

Once these 3 conditions are satisfied by the root server, we proceed with the TLD Server and similarly dive into the hierarchy. We perform the same process for their validation as for the root server and continue the process of DNS Resolving. This process of DNSSEC is similar to the basic DNS resolving, but has these verifications at every stage of the DNSSEC workflow.

After resolving the root server, if in any iteration we receive an "A" type record in the DSRecord Response's Answer section (given the zone is verified), we return that value back to the user. If the Answer section is empty, we further check the Additional Section.
If we obtain IPV4s in the Additional Section of the DSRecord Response body, we iterate through them and feed these IPs back to the iterative GetResponse() function. If the Additional Section is empty, we collect the new NS domain names from the Authority section and feed them back to the Main_Dig function recursively until it is resolved.

This is the entire workflow of my DNSSEC Implementation along with the explanations of the functions used.