The programming language used is python.
The code for the first question is named (Final).
The Stego-image is RGB image of flower and downloaded with the dimensions of 120*180 pixels.

The code read the image from its directory and save it as numpy array named (nd) with 3 dimensions (120*180*3). Which contains the data of the cover-image
The secret message is in the same directory and is a (.txt) file named (message).

THE ALGORITHM:

The text file is first converted into a string of bits according to the ASCI table for each character of the message.
The result for the previous step is a string of bits.
This string is then converted to an integer list named (bits) with a value:
255 represents 1 and 254 represents 0.

The message is divided into four chunks according to the number of characters present in the secret message.
If the number of characters is not divisible by four, there will be 5 chunks, where the last section will contain the remaining characters.
Each character is represented by 8 bits, and the space is a character
Ex: “Hello our friends”
17 chars → 4 chunks of 4 chars each chunk 8*4=32bits
1 chunk of 1 char 8 bits

Ex: “Hello my friends”
16 chars → 4 chunks of 4 chars each chunk 8*4=32bits

After that, each chunk placed on a specific row of the image, starting from left to right, in RGB order for each pixel
This means that for the second example each chunk will need 32 bits which is 11 pixels 10 with (RGB) and the last one (RG) in the following order
R_1 G_1 B_1 R_2 G_2 B_2 R_3 G_3 B_3 R_4 G_4 B_4 R_5 G_5 B_5 R_6 G_6 B_6 R_7 G_7 B_7 R_8 G_8 B_8 R_9 G_9 B_9 R_{10} G_{10} B_{10} R_{11} G_{11}

The method in which we insert the bit in the (R or G or B) of the pixel is as following:

* Assume we deal with the color R and we want to put 0 or 1 in the LSB bit
* We previously saved the 0 as 254 and 1 as 255
* We get the value of the LSB by taking the reminder of dividing the R value by 2 (if the reminder is 0→ LSB=0 else LSB=1)

- If we want to insert 1(255 in the bits array) and the value of the LSB is 0 we increase the value of color by one.
- Otherwise we do (AND logic operation) between the color and the inserted bit which is (255 or 254)
-

Here are some examples

Insert 1 to 26	20+1=21	LSB becomes 1
Insert 0 to the color 27	254&27=26	LSB becomes 0
Insert 1 to color 27	255&27=27	LSB becomes 1
Insert 0 to color 26	255&26=26	LSB becomes 0

Now:
Each chunk is placed in a row of the image
The first row is number 0
The second one is with offset (x rows) from the previous one
x= (number of rows-1) /5 where 5 is the maximum possible number of chunks
we did this to make sure the chunk will be distributed in the image and so will not affect its quality.
The modified (nd) array is then converted to an image named “updated” will be saved in the same directory.

In the decryption:
First, we must know the number of chars in the secret_message
Second, we can know the number of chunks and the number of bits in each chunk.
Third, we can get them from the (nd) array.
The chunks which we will get is a list of 1's and 0's we convert each 8 bits to a decimal value and then we convert this value to a char depending on the ASKI table.
We insert these chars in a string to get the message.
For the secret message (hi I am Iman and I like flowers)

The original Image

The Image after hiding the data:

Question 2:

The python code is named (Ques2)
The hash function used do the following.
Get the sum of all LSB of the message chars
Add them to the number of chars to get a decimal value
Now to make the process easy we represent the result in 8 binary bits
We apply the previous algorithm to hide the information
There is only one byte so the number of chunks is 1 and will be hidden in the first three pixels of the first row.

For the secret message: "hi I am Iman and I like flowers"
0+1+0+1+0+1+1+0+1+1+1+0+0+1+0+0+0+1+0+0+1+1+1+0+0+0+1+1+1+0+1+31=47.
The hash value is 47 and the binary list is [254, 254, 255, 254, 255, 255, 255, 255].