Function Name: `secretAdmirer`

Inputs:

1. (*char*) 1xN string containing an encoded message from your secret admirer
2. (*double*) 1xN vector representing meaningfulness (on a scale from 1-10)

Outputs:

1. (*char*) Meaningful message from your secret admirer

Topics: (*masking*), (*strings and vectors*)

Background:

Is the same person the first to like every Instagram post and to watch your Snapchat story? Do you feel like you bump into them more than you should? Your secret admirer has been waiting for the perfect time to confess their love, and, lucky for them, Valentine's day is coming up. You begin finding notes in your locker (yes, this is definitely a middle school situation), but they don't make sense! This person must be really nervous. Lucky for you, you can decode the messages on MATLAB!

Function Description:

Obviously, your secret admirer means the wonderful things left in the messages. However, because of their nerves, each message also has additional characters mixed in. You need to find only the characters that were written meaningfully in order to read the real message. You are given a coded message (input 1) and a vector of doubles indicating on a scale of 1-10 how much each corresponding character was meant (input 2). If the character is written with a meaning greater than or equal to 6, then keep the character. Otherwise, do not include it in the final message. Lastly, make the whole message lowercase. Your output should be the meaningful message from your secret admirer.

Example:

```
encodedMessage = '!IE Qlivke y?ou^'
meaningVec = [4 7 3 10 1 9 8 2 8 6 9 7 3 8 6 1]
message = secretAdmirer(encodedMessage, meaningVec)
message = 'i like you'
```

Notes:

- The inputs are guaranteed to be the same length.
- The second input will only contain positive integer values
- There will not be special characters in the final string.

Hints:

- Start by making a logical vector.

Function Name: `cupidStats`

Inputs:

1. (*double*) 1xN vector of doubles containing MATLAB skill levels, ranging from 1371-1738
2. (*double*) 1xN vector of doubles containing numbers of seconds to cook Minute Rice™, ranging from 30-90
3. (*double*) 1xN vector of doubles containing Guitar Hero high scores, ranging from 0-985206

Outputs:

1. (*double*) The number corresponding to your love match

Topics: (*masking*), (*comparative operators*), (*logical operators*)

Banned Functions:

`find`

Background:

Come on down! You're the next contestant on Cupid's MATchLAB!!! Bachelor who? A brand new game show designed to find your perfect match, 5 lovers sit behind closed doors awaiting your decision. BUT! You can't pick just anybody! It's 2021 and the bar is pretty high, so you need to find out which suitor meets your rigorous standards. Luckily, we know the three most important facts about each potential lover: their MATLAB skill rating on a scale from 1371 to 1738, the number of seconds it takes them to cook Minute Rice™ between 30 and 90, and their high score on the Guitar Hero 3: Legends of Rock bonus song "Through the Fire and Flames" by DragonForce up to 985,206 points (highest score ever achieved, by Danny Johnson in 2009). Whew, thank goodness you have MATLAB to help you find your soulmate!

Function Description:

The three input vectors will each contain 5 values, one for each lover, with the lovers corresponding to their index position. Using these vectors, apply masks corresponding to the thresholds your lover needs to meet to have a chance at your heart. MATLAB skill (input 1) should be at least 1700 for obvious reasons. Minute rice cooking time (input 2) must be less than 60 seconds in order to demonstrate cooking aptitude. Guitar Hero high score (input 3) needs to be higher than 100,000 to maximize relationship potential. Finally, output the index of the lover who fits all three criteria!

Example:

```
matlabSkill = [1400 1500 1600 1700 1730]
riceTime = [90 80 70 60 50]
ghHighScore = [10 100 1000 10000 100001]
match = cupidStats(matlabSkill, riceTime, ghHighScore)
match = 5
```

Notes:

- The inputs are all guaranteed to be length 5.
- It is guaranteed that only one lover meets all qualifications.
- All stated input ranges are inclusive.

Hints:

- Think about how to get the number of your lover from a logical vector! What could you apply the final mask to?

Function Name: `flowerPicking`

Inputs:

1. (*char*) The type of flower to pick
2. (*char*) 1XN string representing an assortment of various flowers
3. (*double*) 1xN vector that holds how many days each flower has been stored

Outputs:

1. (*char*) A statement on the type of flower and how many you picked
2. (*double*) How many of those flowers were unwilted

Banned Functions:

`strfind`

Topics: (*masking*), (*strings and vectors*)

Background:

Valentine's day is coming up and you are trying to secure a date! You have been following your crush (not creepily) to figure out what type of flowers they would like and alas you get the answer! With this information you go to the flower shop just to be met with so many different options it would take forever to find all the right ones, but luckily you remembered to bring your laptop so MATLAB can save the day!

Function Description:

The desired flower is given as a single letter: ‘r’(rose), ‘t’(tulip), ‘d’(daisy), ‘p’(petunia), and ‘c’(carnation) (input 1) . You are also provided with a vector representing the flowers that exist in the flower shop (input 2). Find how many desired flowers there are in the shop, and output a sentence in the following format (output 1):

```
'There are/is <number of the desired flower> of the right flower(s) in the flower shop'
```

Then determine how many of the desired flowers are not wilted by identifying the corresponding days of storage (input 3). Flowers are wilted if they have been stored for 15 days or more. Output 2 should be the number of unwilted flowers.

Example:

```
assortment = 'rrtcpdttdpprtd'
stored = [12, 1, 17, 25, 6, 4, 34, 11, 2, 11, 1, 45, 7, 15]
[str, num] = flowerPicking('r', assortment, stored)
str = 'There are/is 3 of the right flower(s) in the flower
          shop'
Num = 2
```

Notes:

- The vector of days stored will always be the same length as the original flower shop assortment vector.
- The desired flowers will always be lowercase

Function Name: `sweetHeart`

Inputs:

1. (*char*) A string of chars containing the first part of the broken Sweetheart message
2. (*char*) 1x4*N character vector of the words 'Yay' and 'Nay'
3. (*char*) 1x4*N character vector containing potential words for the missing part of the Sweetheart message

Outputs:

1. (*char*) The message of the fixed Sweetheart

Topics: (*strings and vectors*), (*masking*), (*concatenation*)

Background:

It's Valentine's Day in elementary school, and you just got home with your prettily-crafted shoebox that's stuffed to the brim with candies from your friends and classmates. As you dump out the contents of your makeshift mailbox, you stumble across a box of your favorite festive candy: Sweethearts! Eager to see what messages are printed on the cute little treats, you rip open the box and pour out all of the Sweethearts onto your wooden floor, but you realize a second too late that you were too rough with the candies when they break into smithereens upon hitting the floor! You pick up a few pieces and attempt to put them back together, but no matter how hard you try, you just can't find a perfect match between any of the pieces that makes grammatical sense. Disheartened by this catastrophe, you turn to your trusty and loyal valentine, MATLAB, to piece together the mystery of what messages were written on the Sweethearts.

Function Description:

Given the inputs described above, write a function that performs the following operations:
First, find all positions of the word 'Yay' in the second input. The words in the second input will always be 'Yay' and 'Nay' in this format.
Next, using the positions found in the previous step, find the corresponding words in the third input. If 'Yay' shows up as the second word in input 2, then the corresponding word will be the second **full word** in input 3 (not the second index of input 3). All words in the third input will be 3 chars long, and there will be a space after each word, including the last one.
Finally, concatenate the first input with the words found in step 2 to output the full message of the fixed Sweetheart. Remember to put a space between the first input and the words found in step 2 in your final output, and make sure that the end of your fixed Sweetheart doesn't have an empty space!

Example:

```
brokenMessage = 'BE'
str = 'Yay Nay Nay Nay Nay Nay Nay Yay '
```

```
words = 'MIN HEY THE RE! HRU DOI NG? E<3
[fixedMessage] = sweetHeart(brokenMessage, str, words)
fixedMessage = 'BE MIN E<3'
```

Notes:

- It is possible for you to have more than one instance of ‘Yay’ in the second input.
- Each ‘Yay’ and ‘Nay’ will be separated by a space, with an extra trailing space at the end of the string vector.
- The second and third vectors will always be the same length.

Hints:

- Instead of trying to figure out how to do everything all at once, solve this problem step-by-step.
- The functions zeros() and ones() may be useful in solving this problem.

Function Name: `suitorCompatibility`

Inputs:

1. (*char*) 1x7*N string of chars containing the suitors' names
2. (*double*) 1xN vector containing the suitors' matlab skill levels out of 10

Outputs:

1. (*char*) String announcing the ultimate suitors

Banned Functions:

`repelem()`

Topics: (*masking*), (*strings), (vectors*)

Background:

Being the MATLAB aficionado that you are, Matlabbers from every corner of the world have been lining up all year to be your Valentine. But, you can't compromise for just anyone. After several rounds of intense competition, you've managed to narrow your potential suitors down to the final few. Now, it's time for the final most important test of compatibility. You need to spend the special day with only the gifted few who know bae just as well as you do. Not so soon though! Angry that they didn't make it this far into the courtship contest, someone messed up the order of suitors so they couldn't be matched. Who else could save the day and help you find the gifted few three than bae itself?

Function Description:

Given a list of names in no particular order, rearrange the names alphabetically and then get out the three suitors receiving a MATLAB skill level above a 7. The skill levels only match the names once they are in alphabetical order. Once you have the compatible suitors, output the names in the following format:

```
'And the ultimate suitors are... <name 1> <name2> <name3> !!!'
```

Example:

```
names = 'nikita johnny divesh oliver rohini siddhu '
skill = [9 8 3 6 10 4]
winners = suitorCompatibility(names, skill)
winners =
     'And the ultimate suitors are... divesh johnny rohini !!!'
```

Notes:

- All the names will always be 6 letters long and be separated by a single space
- There will be an extra space at the end of the names string
- There will be only ever by 3 winners each time
- Every name will start with a different letter

Hints:

- The colon operator will be useful to alphabetize the names