Function Name: `teachersPet`

Inputs:

1. (*double*) A 1xN vector of student grades
2. (*char*) A 1xN string of first initials corresponding to the grades
3. (*char*) A 1xN string of last initials corresponding to the grades

Outputs:

1. (*char*) A 1x(3*N-1) string of first and last initials sorted by corresponding grade

Function Description:

You try to get on your professor's good side by mentioning your awesome MATLAB skills. Unfortunately, it backfires because he then asks you to write a function to sort the first and last initials of the students in the class according to their grades.

Your professor gives you a vector of the student grades, a string of first initials corresponding to the grades, and a string of last initials corresponding to the grades. Given these inputs, write a function that outputs a string with the combined first and last initials of every student, with spaces in between, and sorted by the students' grades in descending order. To clarify, an output string might look like this:

```
'RC RW JM LT BS'
```

where the student `RC` had the highest grade, and the student `BS` had the lowest. Note that there should be spaces in between the initials of each student but not after the last student.

Notes:

- The `sort()` function will take care of sorting equivalent grades.

Hints:

- Consider the use of the second input and second output of the `sort()` function

Function Name: `letterWeave`

Inputs:

1. *(char)* A 1xN string
2. *(char)* Another 1xN string
3. *(char)* A 1xM string

Outputs:

1. *(char)* A 1x(2*N+M) string combining the input strings

Function Description:

In order to communicate secretly with your friends, you come up with a method of obfuscating your messages with each other. However, it is tedious to decode the messages manually so you write a MATLAB function to do it for you. Here are the rules for decoding.

A single message is broken into 3 components. The first two components will always be the same length and should be "woven" together such that the characters in the first string go in the odd indices of the new string and the characters in the second string go in the even indices. For example, given the two strings `'EAPESRN'` and `'XML TIG'` the string created by weaving the strings together will be `'EXAMPLE STRING'`. Once you have created this string, you will split it in half and concatenate the first half onto the front of the reversed third string and concatenate the second half onto the end of the reversed third string. So continuing with the same example, if the third string was `'TXET '` then the final string would be `'EXAMPLE TEXT STRING'`.

Hints:

- Think about using the colon operator to create an index vector.
- For even more funzies, you could write the inverse of this function (a function that takes a string and encodes it). Then you could actually use it to communicate!

Function Name: `compareContour`

Inputs:

1. *(double)* A 1xN vector
2. *(double)* Another 1xN vector

Outputs:

1. *(logical)* Whether or not the "contour" of the two vectors is the same

Function Description:

For the sake of this problem, we will define the *contour* of a vector to be whether adjacent elements of the vector are increasing or decreasing. Therefore, any vector defines a sequence of up and down contours based on the difference between adjacent values. The contour of two vectors is the same if they both have the same pattern of increasing and decreasing elements. For the contour to be the same, the vectors DO NOT have to have the same values, nor does the amount by which elements increase or decrease have to be the same; only the pattern of increasing and decreasing matters.

Consider the following vectors

```
v1 = [2, 4, 3]
v2 = [-2, 10, 8]
```

These two vectors have the same contour because they both follow the pattern [increasing, decreasing]. Now consider these two vectors:

```
v3 = [5, 3, 1]
v4 = [1, 4, 1]
```

These vectors do not have the same contour. The first one follows the pattern [decreasing, decreasing] while the second one follows the pattern [increasing, decreasing]

The function should output true if and only if the two vectors have the same "contour".

Notes:

- The `diff()` and `abs()` and `isequaln()` functions will be useful.
- Both vectors will always be the same length.
- A single number has the same contour as any other single number.
- **Make sure that your output is class logical and not double. You will get the problem wrong if you output a double 0 instead of a logical false.**

Hints:

- If N is a number, the sign of N can be found using the formula: $sign\ N = \frac{N}{|N|}$

Function Name: `weeklyCalendar`

Inputs:

1. *(char)* A string representing 7 days of the week (eg. `'N M T W R F S'`)
2. *(double)* A vector representing 7 dates of the week (eg. `[4 5 6 8 7 9 10]`)
3. (double) A positive or negative number of days

Outputs:

1. *(char)* A string representing 7 days of the new week
2. *(double)* A vector representing 7 dates of the new week

Function Description:

It's fairly simple to determine what the date was or will be in a certain number of days. However, it's a little more challenging to figure out what day of the week that new date corresponds to. Thankfully we have MATLAB to depend on for this task.

Write a function that takes in an input string that represents the current *days* of the week (`'N'` = Sunday, `'R'` = Thursday) and a vector that represents the current *dates* of that week. The function will also take in a third input that represents the number of days into the future or past we are shifting the weekly calendar. Using this information, your function should be able to determine the new vector of dates for the new week, in addition to the correct order of days that correspond to these new dates. You should also be able to account for whether the dates go into a new month.

Example:

Given:

```
days  = 'M T W R F S N'
dates = [24 25 26 27 28 29 30]
shift = 5
```

And that the function is called:

```
>> [newDays, newDates] = weeklyCalendar(days, dates, shift)
```

Then the outputs would be:

```
newDays  = 'S N M T W R F'
newDates = [29 30 1 2 3 4 5]
```

Notes:

- Assume that there are always 30 days in a month.
- Remember that there are 7 days in a week.
- Letters that indicate the days of the week are separated by spaces and should be included in your output as well.