Exercise 5:

1. **Summary**

We created the pattern `" "+$` to recognize one or more whitespace characters at the end of each line and the pattern `^" "+` to recognize one or more whitespace characters at the beginning of each line. Then, we used `" "{2,}{printf(" ");}` to replace two or more whitespace characters with a single whitespace character.

On the other hand, to eliminate all empty lines of text, we used the patterns: `^\n+` (those that start with one or more line breaks) and `\n+$` (those that end with one or more line breaks).

2. **Tests**

For the input " Esto es un fichero de texto ", we obtained the output "Esto es un fichero de texto".

For the input "para ser formateado ", we obtained the output "para ser formateado".

For the input:

At first, we only used the pattern `^\n+`, which did not recognize the blank text lines; however, after several tests, we realized that it was necessary to use the following pattern: `\n+$`.

Finally, we obtained the correct output:

Exercise 6:

1. **Summary**

We defined and initialized to zero the following integer variables: TL (total lines), TC (total characters excluding line breaks), TS (total lines containing any punctuation mark), TV (total lines starting with a vowel), TB (total blank lines).

We created the pattern `^" "*[aeiouAEIOU]{TV++; TC+=yyleng;}` to recognize the number of lines that start with a vowel (uppercase or lowercase), meaning lines whose first non-whitespace character is a vowel. We used `yyleng` to know the length of the last captured pattern and added it to the total character count; otherwise, it would be impossible to know the exact number of whitespace characters at the beginning of a line, as there can be 0 or more. For the total lines starting with a vowel, we know that it recognizes a vowel, so we add 1 to TV.

The pattern `^\n{TL++; TB++;}` is used to count the blank lines. Therefore, we add 1 to the variables TL and TB each time a line starts with a line break.

The pattern `\n{TL++;}` is used to count line breaks. That is why we add 1 to TL.

The pattern `^[" "+\n]{TL++; TB++; TC+=yyleng-1;}` is used to detect lines formed solely by whitespace characters. Thus, we add 1 to the variables TL and TB. We added the integer

`yyleng`, which is the length of characters of the recognized pattern, to TC, and we subtracted the `\n` character because the TC variable does not account for it.

The pattern `[.,;:].*\n {TS++; TC+=yyleng-1; TL++;}` recognizes the number of lines that contain a punctuation mark (,.;:) and therefore, we added 1 to TS and TL. We used the TC variable as in the previous case.

The pattern `.{TC++;}` is used to analyze the remaining characters that have not been recognized (any character except `\n`). Thus, we add 1 to TC each time any character appears.

2. **Tests**

For the input `^" "+\n`, we obtained an erroneous output. Thus, we noted the importance of brackets and with the input `^[ " "+\n ]`, we found the correct solution.

For the input provided in the statement:

we get the following output:

Exercise 7:

1. **Summary**

We applied `egrep '^[0-9].*[0-9]$' t.txt` to recognize the lines of a file named `t.txt` that start and end with a digit.

We applied `egrep '^[^0-9]*[^0-9]+[^0-9]*$' t.txt` to display the lines from `t.txt` that do not contain digits.

We applied `egrep '[02468][^0-9]+|[02468]$' t.txt` to recognize the lines in `t.txt` that contain an even number in decimal.

2. **Tests**

At first, we tested with the same regular expressions but without writing `$` at the end of them, and we did not get the results we were looking for.

But finally, we managed to obtain for the following input (file "t.txt"):

```

2hola, que tal estas? 315

estoy 42en casa

35de la familia Costa

hasta las diez

besos

saludos3
```

27 15

8

```

The output for the first `egrep` expression:

```

2hola, que tal estas? 315

27 15

```

The output corresponding to the second `egrep` expression:

```

hasta las diez

besos

```

And the output of the third expression:

```

2hola, que tal estas? 315

estoy 42en casa

8

```

### Exercise 8:

1. **Summary**

We created the pattern `I(E*NE*NE*NE*NE*)+F|IE*F` to recognize the sequence that has a number of N's that is a multiple of 4, with ' - - ' placed at the beginning and the end of the string.

We used `I(E*NE*NE*)F|I(E*NE*NE*NE*NE*NE*NE*(NE*NE*NE*NE*)*)+F` to detect if the string leaves the robot in a position (n, m) with m being an even number that is not a multiple of 4 (an even number of N's that is not a multiple of 4); a ' – ' will be placed at the start and end of the string.

We used `IE*NE*(NE*NE*)*F` to write a ' ' at the beginning and end of the string if it leaves the robot in a position (n, m) with m being an odd number (odd number of N's).
```

2. **Tests**

At first, we performed the exercise focusing on the number of E's. But since we noticed that we were getting correct outputs sometimes and sometimes not, we reviewed the patterns several times, as well as the statement of the exercise. Finally, we observed that the error was a misinterpretation of the statement, since instead of counting the number of E's (as we had been doing), we needed to count the number of N's.

We also encountered issues because we did not realize the existence of the case of 0, which is a multiple of 4, and the pattern `I(E*NE*NE*NE*NE*)+F` did not reach its target by itself; thus, we added `| IE*F` for the case where there were zero N's.

After trying several times with the following patterns:

I((NE*|E*N)(NE*|E*N))F|I((NE*|E*N)(NE*|E*N)(NE*|E*N)(NE*|E*N)(NE*|E*N)(NE*|E*N)E*(NE*NE*NE*N)*)+F

I((NE*|E*N)(NE*|E*N)(NE*|E*N)(NE*|E*N)E*)+F|IE*F

IE*NE*F|I(E*N(NE*|E*N)E*NE*(NE*NE)*)+F

We realized that it worked the same while simplifying the expressions much more.

After several attempts, for the input given in the statement, we obtained the correct output:

Entrada:

```
querido francisco:
me puedes marcar como hemos quedado
las cadenas
INENEF, IENNENENF,
INENENENENNF
IEEEF INEF
```

Salida:

```
querido francisco:
me puedes marcar como hemos quedado
las cadenas
-INENEF-, --IENNENENF--,
-INENENENENNF-
--IEEEF-- *INEF*
```