Oppg 2

Line one

```
char *line = nullptr;
```

This defines a char pointer that currently points at nothing. Earlier versions of C++ set a nullptr to 0, as that is the one address reserved to point at nothing.

Line two

```
strcpy(line, "Dette er en tekst");
```

Strcpy copies one string from one array to another array

```
char * strcpy ( char * destination, const char * source );
```

Danger

Problem with this code is that the initialisation did not make space in the memory for an entire list. This means that any char past the initial char in this list will override whatever comes next to it, possibly deleting important variables and create an unexpected bug that can be quite inconsistent.

To solve this, replace the first line with this:

```
char line[17];
```

This will initialize enough space for the entire string we're going to copy including the "\0" char that indicates the end of the string. Being sure to initialize enough space for the entire array, means that the compilator can avoid overwriting something important.

Oppg 3

First thing that comes to mind is that the wile loop in line 5 can result in an indefinite loop if there are no 'e' char in the input. To solve this, you could add an or operator in the while statement along with *pointer !="\0". The "\0" should signify the end of a string. Secondly, if the input is longer than 4 letters, we will have the same issue as in the previous task. We could possibly override data in the stack, or accidentally jump out of the allocated memory. In order to fix this, you could use the setw(int length) function from the iomanip library to limit the length of the input.