Exercise 3

Take a look at the files in ex3 directory. These files should be familiar as they implement a linked list with dynamic allocation on the heap. This program does not have any known bugs. But we will use it in this exercise to examine the content of memory a bit more closely.

Compile the code using

```
gcc -g list.c list_test.c -o list_test
```

and then start the program in the `gdb` debugger:

```
gdb ./list_test
```

Let's set the breakpoint to line 17 in `list_test.c` file and then run the program to that line.

```
b list_test.c:17
run
```

The next line contains a call to the `addFront` function. We will step into that function to examine it further: just execute `step` instruction.

The next line to be executed is the call to `malloc` - use `next` to step over this function call. Issue `next` instructions a few more times until you reach line 18

```
(gdb) n
18      (*head) = n;
```

Before we continue, let's figure out the address of memory block that `malloc` just allocated:

```
p n
```

Make a note of this address as the memory address of the first created nodes.

Since the structure of the node contains two pointers, you can also see where they are pointing to:

```
x/2xg n
```

Continue executing the program to step into the next two calls to `add_front` function.

- Note the memory addresses of each of the two created nodes.
- Also note the memory addresses stored in the pointers in each of the nodes

When back in `main()` function, execute the following print instructions:

```
p head
p head->next
p head->next->next
```

What is the relationsship between the memory addresses you noted before and the ones that were just printed?

Now run the the following instructions

```
(gdb) x/2xg head->next
(gdb) p head->next->word
(gdb) p head->next->next
```

Study the memory addresses displayed by all three instructions.

Continue through the rest of this program and experiment with different `gdb` instructions to display memory content.