1 Repeat Implementation:

- My implementation of the repeat command function receives a vector of strings, where each element is a token of the original command passed to the shell delimited by spaces. Then it parses this vector of strings in order to create a different vector that will be passed to my **`ExecSystem`** function. For instance, it transforms **`repeat 5 /usr/bin/xterm`** into **`background /usr/bin/xterm`**. Then it passes the command to **`ExecSystem`** in a for loop that runs n times. With the previously mentioned command, it would pass **`background /usr/bin/xterm`** 5 times to the **`ExecSystem`** function.
- Relevant Functions:
  - **`RepeatedCommand`** (**`repeat`** command)
  - **`ExecSystem`** (**`start`** and **`background`** commands)

2 Dalekall Implementation:

- My implementation of **`dalekall`** is a void function that works on the **`backgroundPid`** vector of integers in the **`Shell`** class. The pid of each process launched in the background is appended to this vector. (Additionally, background processes that are killed via the **`dalek`** command are removed from this vector.) When the **`dalekall`** command is called, it prints the number of processes that will be killed and calls **`dalek`** on the first pid in the vector until the vector's size reaches 0. My implementation works in this manner because when the first item in the vector is killed, it's size is reduced by 1 and the second item in the vector becomes the new first item.
  - **`ExecSystem`** (only relevant in creating **`backgroundpid`**)
  - **`KillSystem`** (**`dalek`** command)
  - **`KillAll`** (**`dalekall`** command)