# Homework 2

In Homework 1, you used shell scripts to create a semblance of a vacuum cleaner. In this homework, please do the same thing, except use C.

- Try the problems given in the lecture. You need not hand anything in for this part.
- Try compiling and running the example C programs. You need not hand anything in for this part. To make use of the `examples.tar.gz` file:
  - `gunzip examples.tar.gz`
  - `tar -xf examples.tar`
- Write vacuum.c which may be compiled by cc -Wall vacuum.c -o vacuum.
- It should take two command line arguments: vacuum qtyrows qtycols.
  Note: your program should take arguments; however, if it does not get two arguments, your program should emit a "usage" message and exit, not issue a core dump. Interpret with: sscanf(argv[1],"%d",&qtyrows); and sscanf(argv[2],"%d",&qtycols);. When you go to the next line, either issue a newline or use fflush(stdout); to push the output to your display.
- Note that `\033` stands for the escape character when inside strings.
- Use `calloc()` to allocate enough memory to hold `qtycols` characters in strings:

```
char **goright;
...
goright= (char **) calloc(qtycols,sizeof(char *));
```

- You'll also need to allocate enough memory to hold each string on a string by string basis: (modify as needed to handle `goleft`, too)

```
vaclen     = 5; /* leave space for the null byte */
goright[0] = (char *) calloc(vaclen,sizeof(char));
strcpy(goright[0],"\\==<");
for(i= 1; i < qtycols; ++i) {
  ++vaclen;
  goright[i]= (char *) calloc(vaclen,sizeof(char));
  sprintf(goright[i]," %s",goright[i-1]);
  }
```

- Use `printf()` to send the strings to the display. You may need to use `fflush(stdout);` to flush the buffered output to the display.
- Like the shell script, you'll need some delay to make the vacuum cleaner appear to move:

```
#include <stdlib.h>
#include <unistd.h>
...
usleep(10000); /* for ten milliseconds */
...
```

- Use the man command to see the manual for various commands and functions. For example: `man 3 printf`.
- Note that "\==<" is incorrect because C tries to interpret "\=" as an escape sequence, and it isn't one. Try escaping the backslash: "\\==<" instead.
- There's no need to use the "shell()" function