Never tell me the odds!

- Han Solo

This is a 120 minutes exam, commencing at 19:00 and finishing at 21:00. (For CSD students it ends at 23:00) You may use your notes, textbook, PDF of textbooks and laptop to access Mimir. You cannot use StackOverflow, Chegg, or any other online sources. Plagiarism checks are in full force. Cheating on the midterm will be severely punished (it will lead to an immediate 'F').

There are a handful of questions on this test. **Read the description carefully and attentively before diving into the code.**

Q1. Mean and Standard Deviation

You are to write a simple C program that reads, from its standard input, a sequence of integers separated by whitespaces (blanks, tabulations or line feeds) and computes both the mean and the standard deviation of these numbers. Recall that, given numbers

$a_0 \ldots a_{n-1}$

The mean is

$$\frac{\sum_{i=0}^{n} a_i}{n}$$

And the standard deviation is:

$$\sqrt{\frac{\sum_{i=0}^{n} a_i^2}{n} - \left(\frac{\sum_{i=0}^{n} a_i}{n}\right)^2}$$

Your program should produce a simple output with the two values. e.g, for the input

```
2 3 4
```

it would produce

```
mu=3.000000 sig=0.816497
```

Q2. We Shall Trim!

In this question you are expected to write a convenience function that *trims* a string by removing all trailing spaces. Namely, once trimmed, the last character of the string is *not* a space. Spaces are considered to be the following characters:

- White spaces ' '
- Tabulations '\t'
- Line feeds '\n'
- Carriage returns '\r'

You *cannot* make use of any standard library function. If you need to introduce auxiliary functions, feel free to implement them and call them. The entire function is just a handful of lines of code.

The API of the function is quite simple

```
char* strip(char* s);
```

Since it takes as input the string s and produces, as output, a pointer to the trimmed string. Note that the trimming is "in-place", i.e., you are not creating a copy of the original, you modify the original in place.

Q3. Did I say Upper Case ?

You are expected to write a function which, when given a string s, creates and returns *a new heap allocated string* that is identical to s in all respects except that every alphabetic character was replaced by its upper-case equivalent.

The function has the simplest protoype

char* toupper(char* s);

And it produces a new string (it leaves s untouched) with a content identical to that of `s`, except that every alphabetical character is replaced by its upper case variant. **You are not supposed to use any library function.** If you need an auxiliary function, you **must** provide it yourself. Note that the newly allocated string has the same length as `s` and is hosted on the system heap.

Q4. Doubly, you said...

In this question, you are to build an ADT for a venerated data structure. The doubly-linked list. You will be working with a list of integers and we supply the test program. The API contains 8 functions to setup and manipulate a doubly linked list. Specifically, there are functions to:

- initialize the list
- clear the list (make it empty, releasing whatever it may contain)
- insert an element at the front of the list
- insert an element at the back of the list
- insert an element right after a given element key that may or may not appear in the list (if it does not appear, add the new element at the end)
- remove an element from the list
- print the list going in the forward direction (front to back)
- print the list going in the backward direction (back to front)

This question will have many test cases to evaluate all the APIs and their interplay. We are providing the test program that you should **not** modify. We are also providing a sample test `(data.txt)` that you can feed to the test program for some modest testing. The automated test are quite a bit more thorough.

To compile the program, simply run:

```
make
```

To test the program simply run

```
dltest < data/data3.txt
```

And it will read its input from `data/data3.txt`.

You should not have crashes of any kind nor any memory leaks. The header file is fixed and should not be modified. You will focus. exclusively on the `dllist.c` file. (The grading suite has a lot more tests. We give you one small one). Note that the testing input is a file with a sequence of *commands*, one per line. each command has 0, 1 or 2 arguments. The commands are as follow

1. `fr <value>` : inserts <value> at the front of the list
2. `bk <value>` : inserts <value> at the back of the list
3. `in <key> <value>` : inserts <value> right behind the first occurrence of <key> in the list.
4. `rl <value>` : removes the first occurrence of <value> from the list.
5. `pf` : print the list in the forward direction.
6. `pb` : print the list in the backward direction.
7. `cl` : clears the list (remove all values held in the list, i.e., the list is empty again).

So the sequence of commands:

```
fr 3
fr 2
fr 1
pf
```

inserts 3 in the front, then 2 in the front, then 1 in the front and prints the list forward.