Deadline: January 29, 2024 at 23:59

Brainfry

The objective of this assignment is to put into practice the learnt assembly concepts and problem-solving skills in order to decompose and implement a complex task (by the standards of assembly programming) while testing your ingenuity. You will be implementing a compiler for an esoteric programming language called Brainfry (reminiscent of a similar, not-so-appropriately-named language).

Before you continue reading

If you have any problems with Themis compilation/runtime errors (all test cases are openly visible), please send an email to the helpdesk as soon as possible.

Description

Brainfry is an esoteric programming language that uses a small set of simple commands to manipulate a virtual "tape" of memory cells. The tape is represented as an array of memory cells, each of which can hold a single data byte. The commands in Brainfry manipulate the tape by moving the "tape head" (which is a pointer to the current memory cell) left or right, incrementing or decrementing the value of the current memory cell, and accepting or outputting the value of the current memory cell.

The commands in Brainfry are:

- `>`: move the tape head to the right.
- `<`: move the tape head to the left.
- `+`: increment the value of the current memory cell.
- `-`: decrement the value of the current memory cell.
- `.`: output the value of the current memory cell as an ASCII character.
- `,`: input a value from the user and store it in the current memory cell.
- `[`: if the value of the current memory cell is 0, jump to the corresponding `]` command.
- `]`: if the value of the current memory cell is not 0, jump to the corresponding `[` command.

All other input characters are ignored. The program written in Brainfry is executed by an interpreter that reads the commands one by one and executes them to manipulate the (contents on the) tape.

Example

```
[
This loop is an "initial comment loop", a simple way of adding a comment
to a BF program such that you don't have to worry about any command
characters. Any ".", ",", "+", "-", "<" and ">" characters are simply
ignored, the "[" and "]" characters just have to be balanced. This
loop and the commands it contains are ignored because the current cell
defaults to a value of 0; the 0 value causes this loop to be skipped.
]

[ This is a comment ]
> ++
[
This is also a comment but
-
The instruction above got executed
]
~
```

You can now look at the second memory register, and you should see the value 0 (`x0000`) in there.

Constraints and simplifications

Usually, Brainfry's tape is 65k (65,000) elements long. However, to accommodate the limitations of LC3, your tape does not have to be longer than 5k, and it's guaranteed that the programs that we submit to your compiler will not go into negative tape indexes.

Additionally, a normal Brainfry program would stop when it reads the EOF (End of File) symbol, however, to simplify this, we have decided to add a `~` character to symbolise "End of Program". What follows after this symbol is the input to the Brainfry program. Note that all programs are built to read deterministic code.

Furthermore, for this assignment, all memory has been set to 0; you do not need to preset it.

Report

Aside from the programming part, you can also get up to two bonus points by writing a short (at most **2** pages) report in LaTeX that should contain the following sections:

- problem analysis;
- algorithm and program design;
- implementation choices.

Note that you cannot turn in a report without having attempted to do the programming part as well. If your program does not pass all the test cases, be sure to include in your report what you have tried, and where you suspect your program might be flawed.