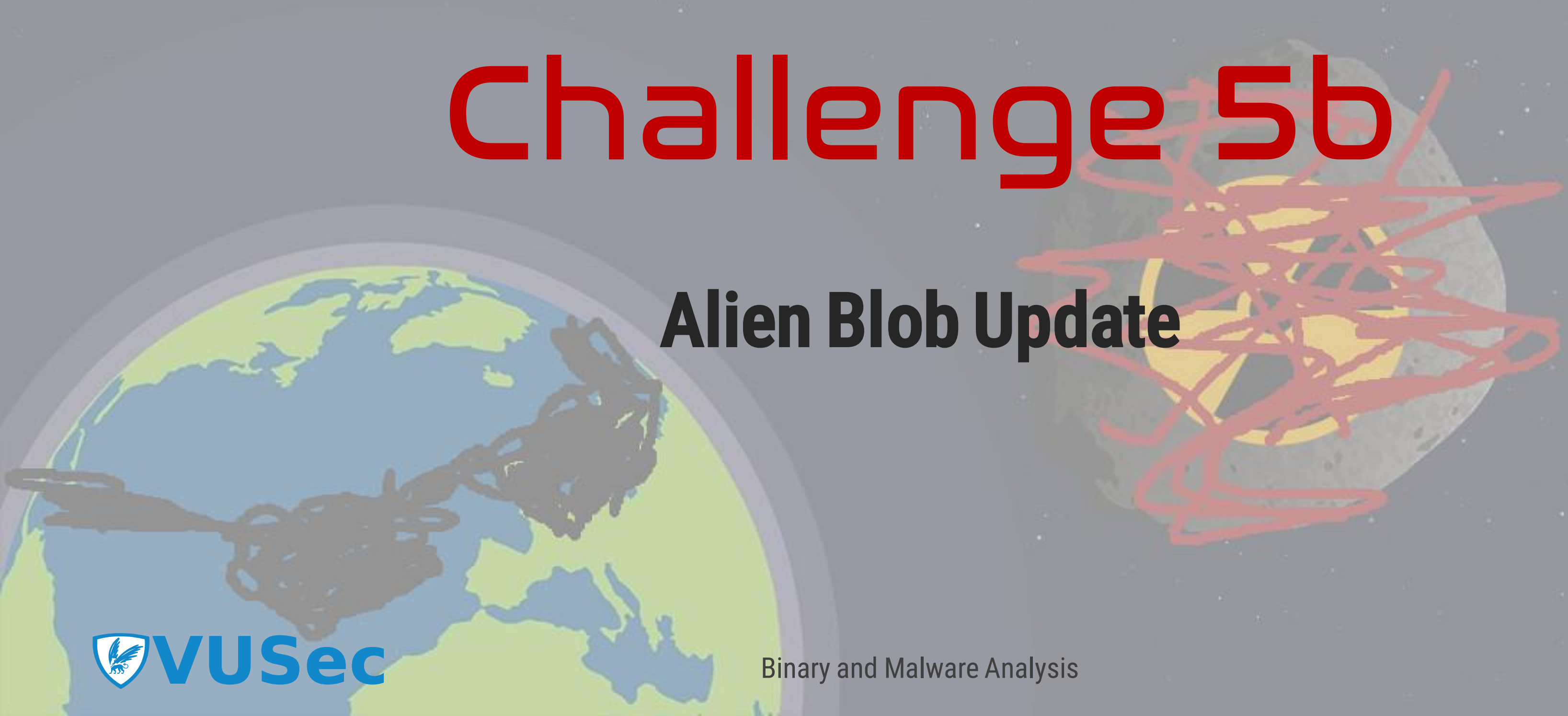
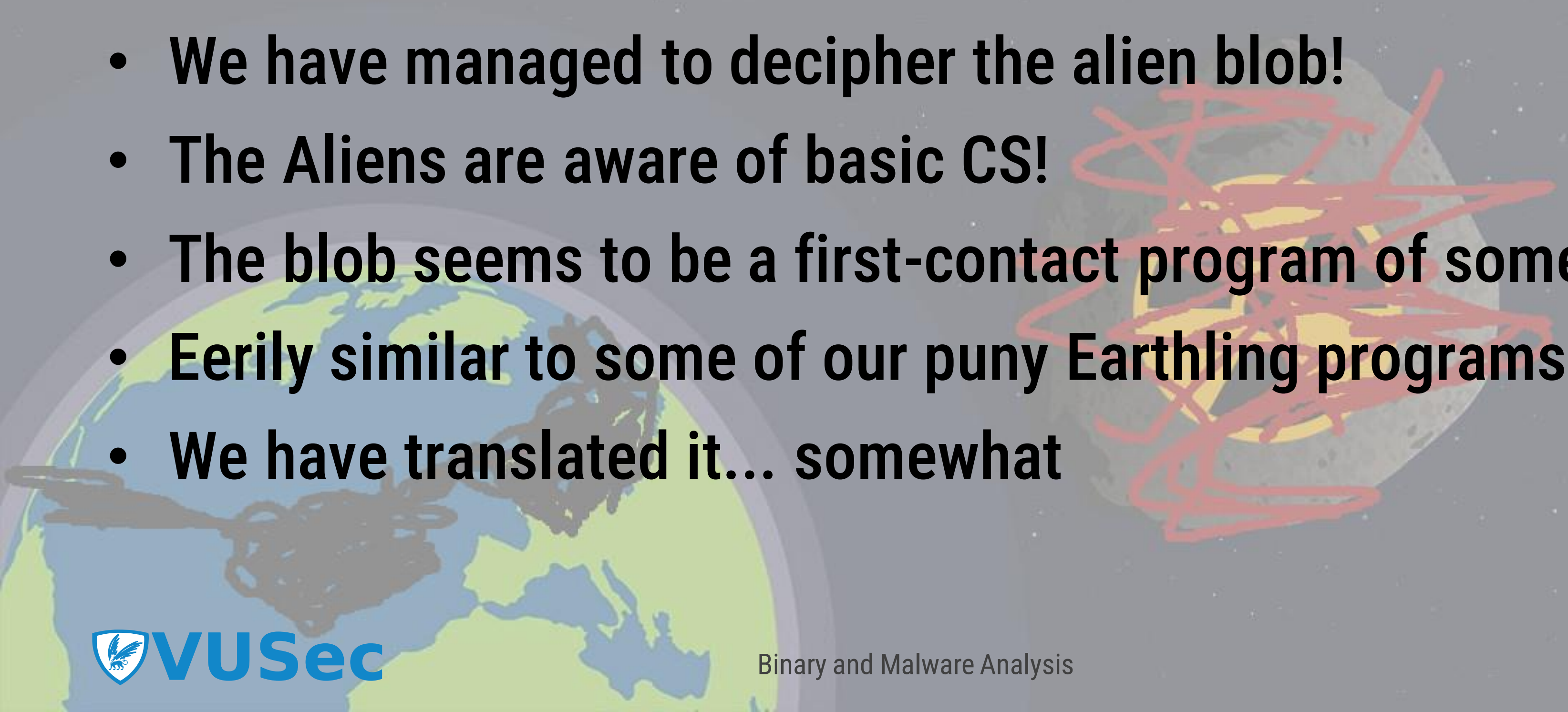


Challenge 5b

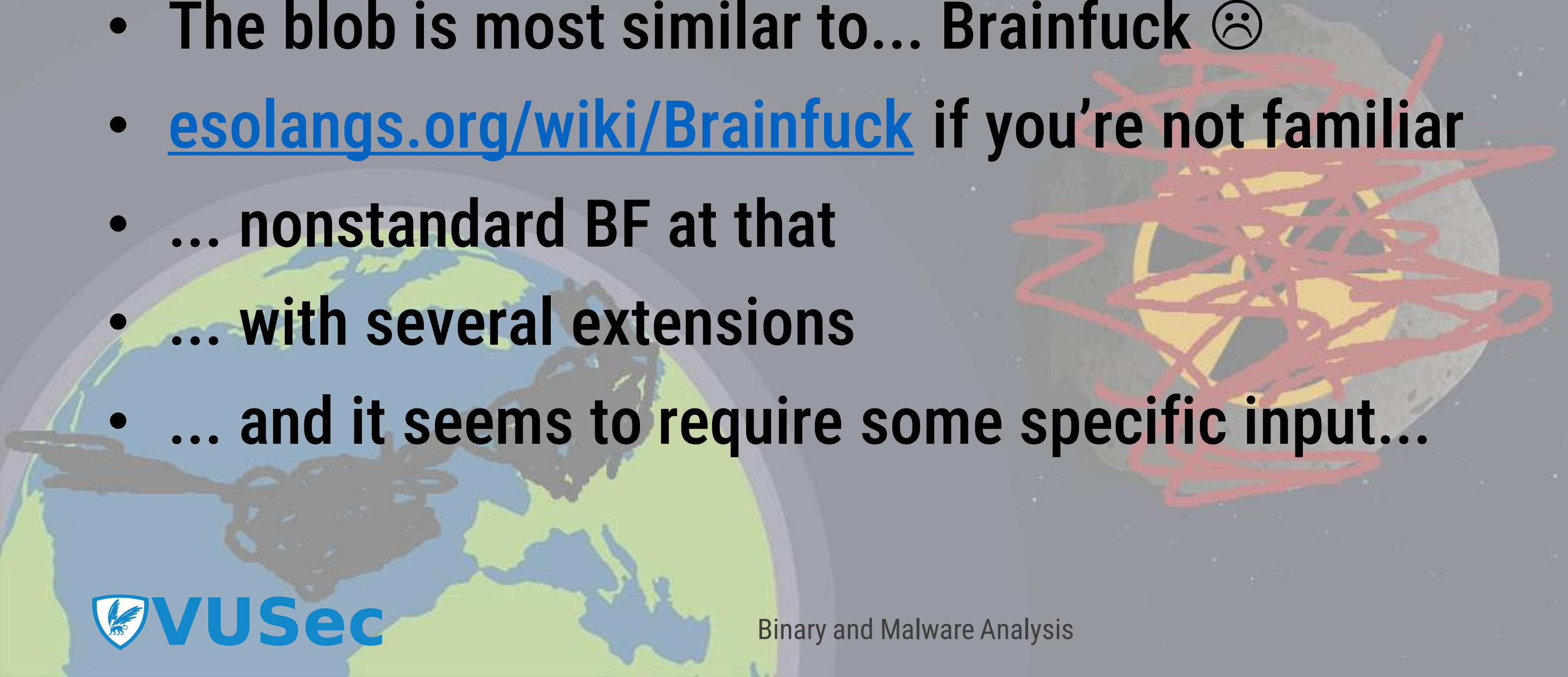
Alien Blob Update



Good News!

- We have managed to decipher the alien blob!
- The Aliens are aware of basic CS!
- The blob seems to be a first-contact program of some sort
- Eerily similar to some of our puny Earthling programs
- We have translated it... somewhat

Bad News ☹



- The blob is most similar to... Brainfuck ☹
- esolangs.org/wiki/Brainfuck if you're not familiar
- ... nonstandard BF at that
- ... with several extensions
- ... and it seems to require some specific input...

Good News Again!

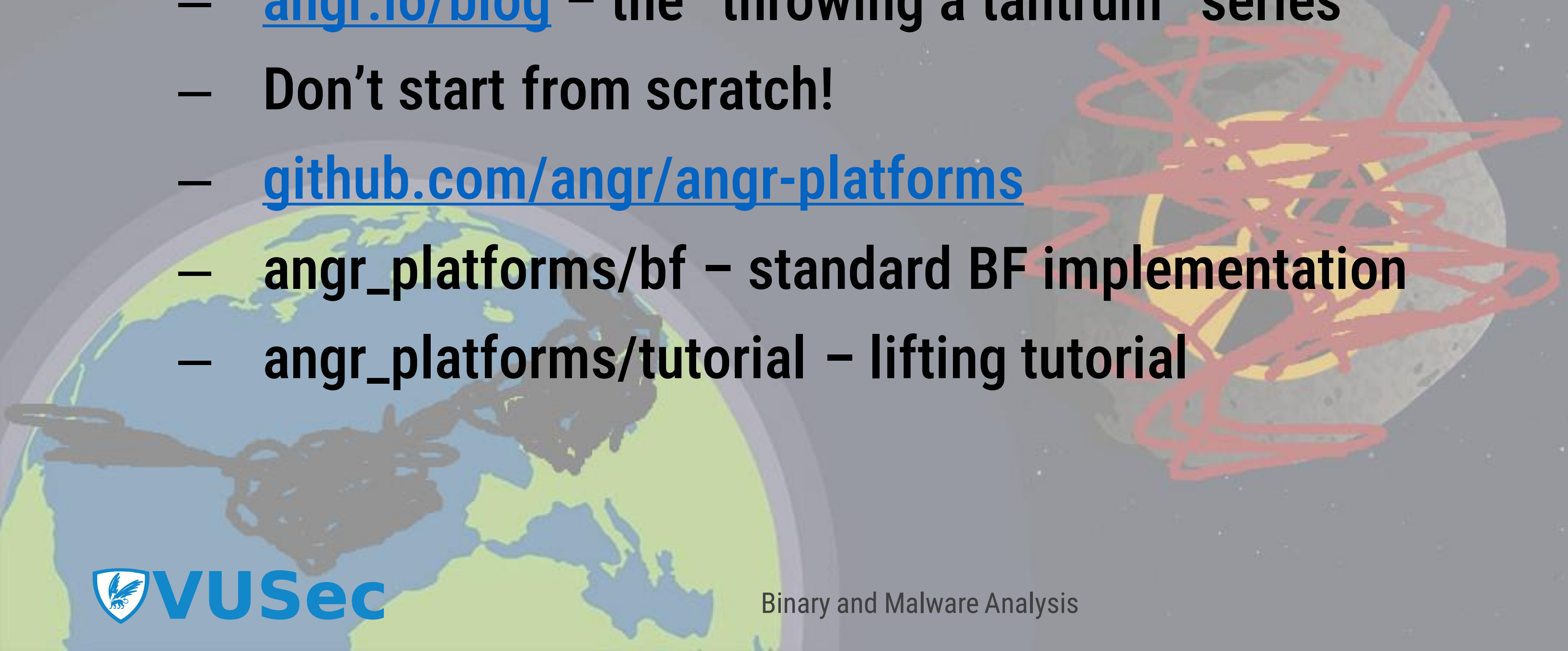
- Mad people have already implemented a lifter for BF!
- We can ~~steal~~ build upon and adapt their work!
- Our xenoprogram experts figured out the Alien BF specs!
- ... not the required inputs though...
- ... nor what it actually *does*...

TASKS

- Write an angr lifter for ABF
- Run the first-contact program through angr
- Find out what inputs it wants and how it reacts to them
- Find out what fist-contact message the aliens have sent

In Detail

- Write an angr lifter for ABF
  - angr.io/blog – the “throwing a tantrum” series
  - Don’t start from scratch!
  - github.com/angr/angr-platforms
  - angr_platforms/bf – standard BF implementation
  - angr_platforms/tutorial – lifting tutorial

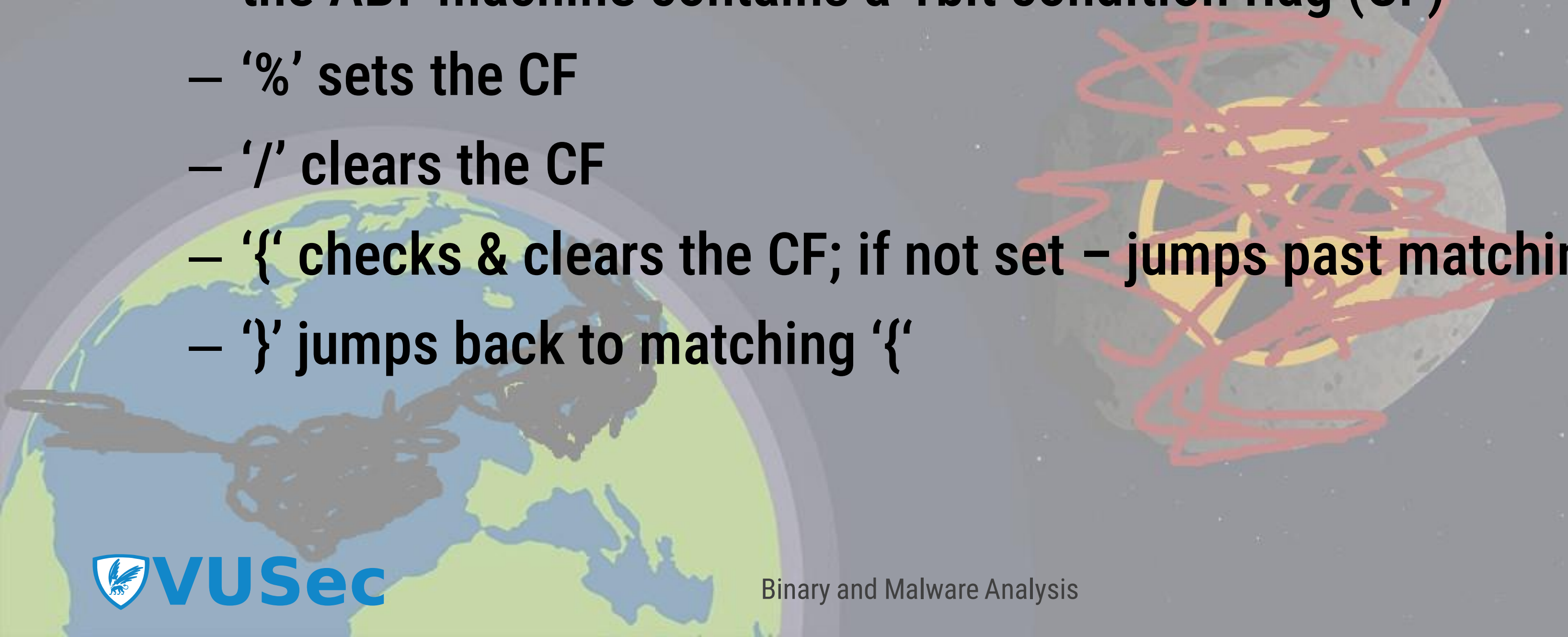


In Detail

- ABF differences to regular BF
  - I/O operators are self-incrementing
    - i.e., they move the current cell to the right
    - ‘.’ and ‘,’ in ABF are equivalent to ‘.>’ and ‘,>’ in BF, respectively
  - Condition flag extension
  - Stack extension

In Detail

- Condition flag extension:
  - the ABF machine contains a 1bit condition flag (CF)
  - ‘%’ sets the CF
  - ‘/’ clears the CF
  - ‘{‘ checks & clears the CF; if not set – jumps past matching ‘}’
  - ‘}’ jumps back to matching ‘{‘

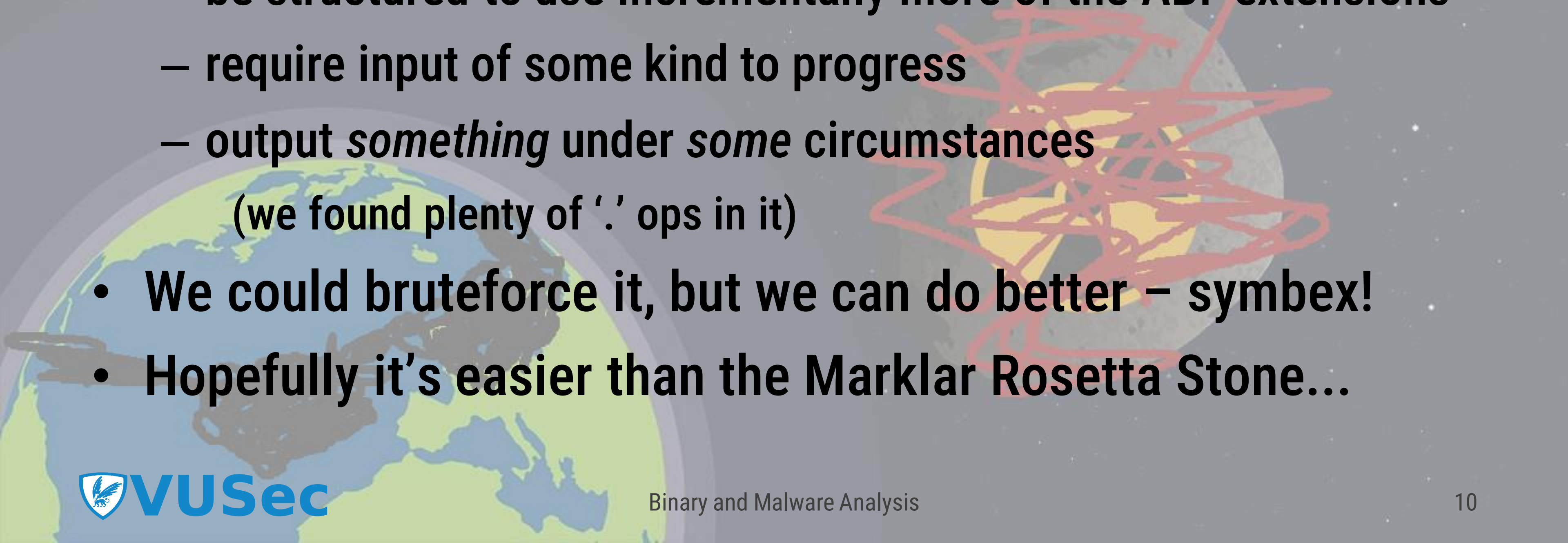


In Detail

- Stack extension:
  - the ABF machine contains a bottomless stack
  - bottomless == a pop from an empty stack yields 0
  - ‘v’ pushes the current cell on the stack
  - ‘^’ pops the top of the stack to the current cell
  - ‘#’ pops the stack and adds it to current cell
  - ‘=‘ pops the stack and subtracts it from current cell

In Detail

- The first contact program seems to:
  - be structured to use incrementally more of the ABF extensions
  - require input of some kind to progress
  - output *something* under *some* circumstances
    (we found plenty of '.' ops in it)
- We could bruteforce it, but we can do better – symbex!
- Hopefully it's easier than the Marklar Rosetta Stone...



Grading

1. Adapt angr_bf's I/O to the alien dialect & recover 1st part – 1p
   - Adapt angr_bf's I/O to the alien dialect (angr_abf) – 0.5p
   - Determine required input & recover 1st part of message – 0.5p
2. Implement the condition flag extension & recover 2nd part – 1p
   - Extend angr_abf to support the condition flag ops – 0.5p
   - Determine required input & recover 2nd part of message – 0.5p
3. Implement all of ABF and recover all the message – 1p
   - Extend angr_abf to support the stack ops – 0.5p
   - Determine required input & recover last part of message – 0.5p

Submission Guidelines

You need to add the following to your challenge 5 **zip** file:

- ‘**angr_abf/***’ – python package implementing your Alien BF lifter for angr
- angr scripts that solve your first-contact program
- ‘**README_ABF**’ describing what you did, how and why

Submission Guidelines

- You will receive your ABF programs by e-mail
- Submission will be through **Canvas** as part of Challenge 5.

GOOD LUCK!