add ebx, 8 ← after each pair

sub ebx, —

(Automated) Memory Management

(read_line)

```
; just-read.snake

(def (our_main _)
  (read_line))
```

```
$ ./just-read.run
10 9 7
(10,(9,(7,false)))
```

```
; simple-read.snake

(def (our_main _)
  (let (line (read_line))
    (fst (snd line))))
```

simple

```
$ ./just-read.run
10 9 7
???
```

```
/* Reads a line of space-separated numbers from the user. Allocates
a pair-based list containing those numbers, and returns a reference
to it, updating the heap pointer appropriately. Pairs are allocated
in order with the list.*/
int* read_line_c(int* heap_start) {
  char* line = NULL;
  size_t size = 0;
  getline(&line, &size, stdin);
  char* tok = strtok(line, " ");
  while(tok != NULL) { ... }
}
```

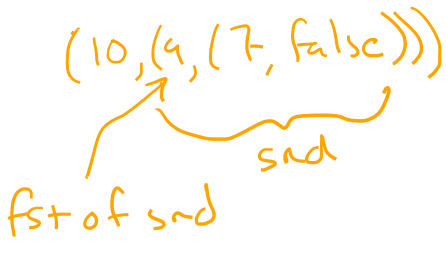
A: 10

B: 9

C: 7

D: Error

E: False



← B: ((10, (90, false))

C: ((10,90), false)

D: Something else

```
; min-max.snake

(def (max lst) ...)  ; return largest num in list
(def (min lst) ...)  ; return smallest num in list
(def (maxmin l)
  (pair (min l) (max l)))
(def (read_maxmin)
  (let (line (read_line))
    (if (== line false) false
        (maxmin line))))
(def (our_main so_far)
  (let (next (print (read_maxmin)))
    (if (== next false)
      so_far
      (let (updated (print (pair next so_far)))
        (our_main updated)))))
```

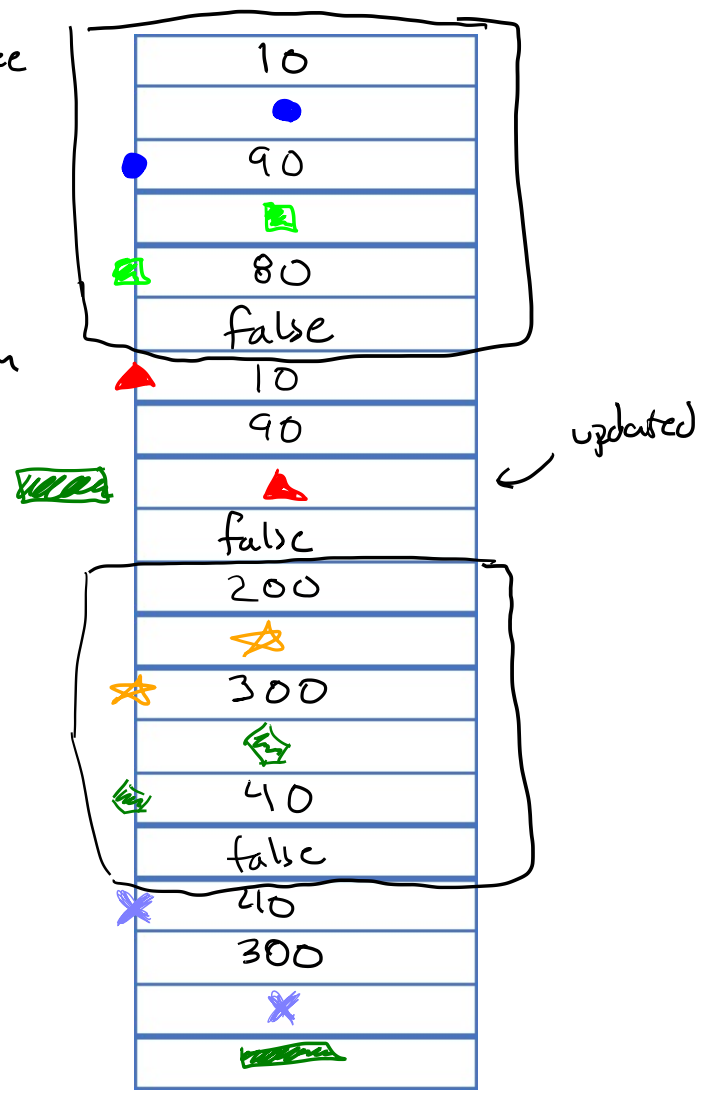
delete/free line

```
$ ./min-max.run
10 90 80
(10,90)
???      ; first updated print
200 300 40
(40,300)
???      ; second updated print
```

((10,90), false)

((40, 300), ((10,90), false))

1. We could delete/free like manually at the right place
2. We could use info about program to know line is OK to delete.



; min-max.snake

```
(def (max lst) ...)  ; return largest num in list
(def (min lst) ...)  ; return smallest num in list
(def (maxmin l)
  (pair (min l) (max l)))
(def (read_maxmin)
  (let (line (read_line))
    (if (== line false) false
        (maxmin line))))
(def (our_main so_far)
  (let (next (print (read_maxmin)))
    (if (== next false)
      so_far
      (let (updated (print (pair next so_far)))
        (our_main updated)))))
```

```
$ ./min-max.run
10 90 80
(10,90)
???      ; first updated print
200 300 40
(40,300)
???      ; second updated print
```