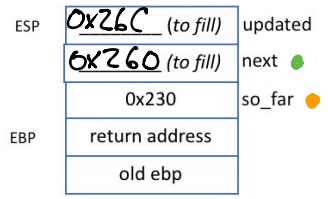
```
; min-max.snake

(def (max lst) ...)  ; return largest
(def (min lst) ...)  ; return smallest
(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)
((10,90),false)
200 300 40
(40,300)
((40,300),((10,90),false))
```



EBP 0x278

0x200	_	10	0x20c
0x20c	_	90	0x218
0x218	_	80	false
0x224	●	10	90
0x230	●	0x224	false
0x23c	_	200	0x248
0x248	_	300	0x254
0x254	_	40	false
0x260	●	40	300
0x26c	●	0x260	0x230
0x278			

mark

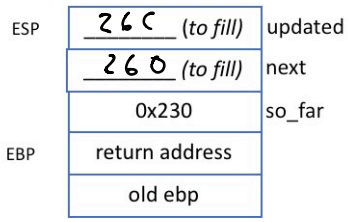
What state is depicted at the start? *(A few local vars are omitted for simplicity)*

A: Right at the beginning of the function body of the 3rd call to our_main

B: Right before the 3rd call to our_main

C: Right at the end of the second call to (read_maxmin)

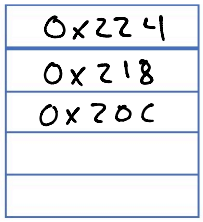
D: Right before printing (read_maxmin) for the third time



EBX 0x278

0x200	_	10	0x20c
0x20c	_	90	0x218
0x218	_	80	false
0x224	●	10	90
0x230	●	0x224	false
0x23c	_	200	0x248
0x248	_	300	0x254
0x254	_	40	false
0x260	●	40	300
0x26c	●	0x260	0x230
0x278			

mark



EBX 0x278

	_	10	0x20c
	_	90	0x218
	_	80	false
->200	10	90	
->20C	0x224	false	
	_	200	0x248
	_	300	0x254
	_	40	false
->218	40	300	
->224	0x260	0x230	

forward

EBX 0x230

_	10	90
_	0x200	false
_	40	300
_	0x218	0x20C

compact