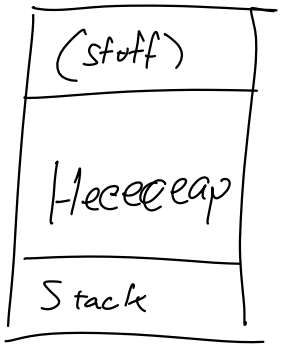
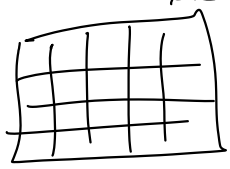
Not Eagle

```
<expr> ::= ...
         | (<expr>, <expr>)
         | fst <expr>
         | snd <expr>
         | ispair(<expr>)
```

Ex. (2, 5)



Pigeonhole Principle

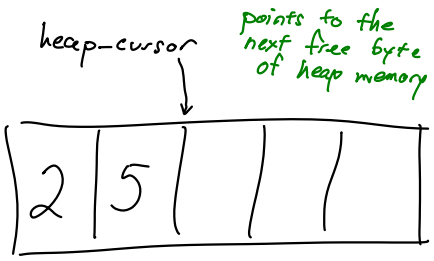


S slots
T things

then at least one slot contains at least $\lceil T/S \rceil$ things

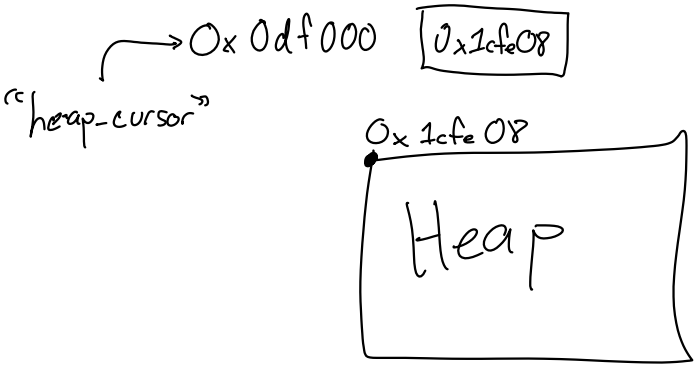
```
let p = (2,5) in
fst p + snd p
```

To initialize heap_cursor, we'll use malloc.
To store heap_cursor, we'll use a global variable in our data section.




```
align 8
section .data
heap_cursor:
    dq 0
section .text
bird_main:
    push rbp
    ⋮
```

```
uint64_t v = bird_main(heap_ptr);
```




```
bird_main:
    push rbp
    mov  rbp, rsp
    sub  rsp, ...
    mov [heap_cursor], rdi
```

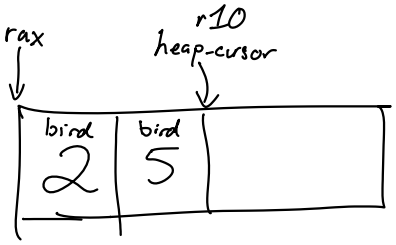
```
mov r10, [heap_cursor]
mov r11, [r10]
```

NotEagle

Write asm for (2, 5)
Assume that heap_cursor is already set up

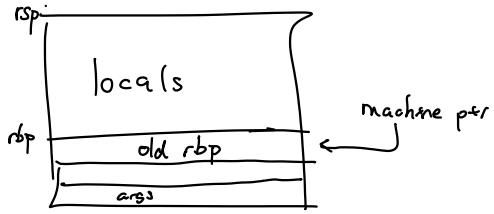
Assembly for (2,5) in NotEagle

```
mov r10, [heap-cursor] ; r10 points to next free mem
mov rax, r10
add r10, 16
mov [heap_cursor], r10
mov [rbp-8], rax   ; allocated space
mov rax, 4
mov r10, [rbp-8]
mov [r10], rax
mov rax, 10
mov r10, [rbp-8]
mov [r10+8], rax
mov rax, [rbp-8]
or rax, 1
```



bool — 0x[b111]FF···FF[1111]
int — 0x ZZZ ····· ZZ[zzz0]
ptr — 0xNNN ····· NN[n001]

machine value = bird value
0xcf1b09 = R(0xcf1b08 as ptr)



Eagle

<expr> ::= ····
| (<expr>, ..., <expr>)
| <expr> [<expr>]

heap objects

Tuples

(1, 2, 3)