Section 4 Slides 86-87

low

3 g:
  y: 8
  c: 64
  return address: f: 13

2 f:
  x: 2
  b: 5
  d: ??? 7 114
  return address: main: 18

main:
  a: ??? 9 114
  return address: OS

high

Section 4 Slide 92

low

sum_first:
  n: 0
  return address: sum_first: 5

sum_first:
  n: 1
  return address: sum_first: 5

sum_first:
  n: 2
  return address: main: 10

main:
  a: ???
  return address: OS

high
# Section 4  Slide 98

```c
int main(void) {
    for (int j = 0; j < 3; ++j) {
        int k = 0;
        k = k + j;
        trace_int(k);
    }
    return 0;
}
```

---

<table>
<thead>
<tr>
<th>j</th>
<th>0</th>
<th>x</th>
<th>2</th>
<th>3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

return address: 05
#include "cs136.h"

int reverse_input_count(int acc) {
    int n = read_int();
    if (n == READ_FAIL) {
        return acc;
    } else {
        int count = reverse_input_count(acc + 1);
        printf("%d \n", n + count);
        return count;
    }
}

int main(void) {
    reverse_input_count(0);
}

Q2b. Write a program that reads in ints, and determines the count of how many numbers were read in. It prints the numbers in reverse order, adding the count to each number.
Outputs: 33
23
13

Inputs:
10, 20, 30.

reverse_input_count:
count: ??
n: READ-FAIL
acc: 3
return address: reverse_input_count: 8

reverse_input_count:
count: ??
n = 30
acc: 2
return address: reverse_input_count: 8

reverse_input_count:
count: ??
n = 20
acc: 1
return address: reverse_input_count: 8

reverse_input_count:
count: ??
n = 10
acc: 0
return address: main: 15

main:
return address: OS