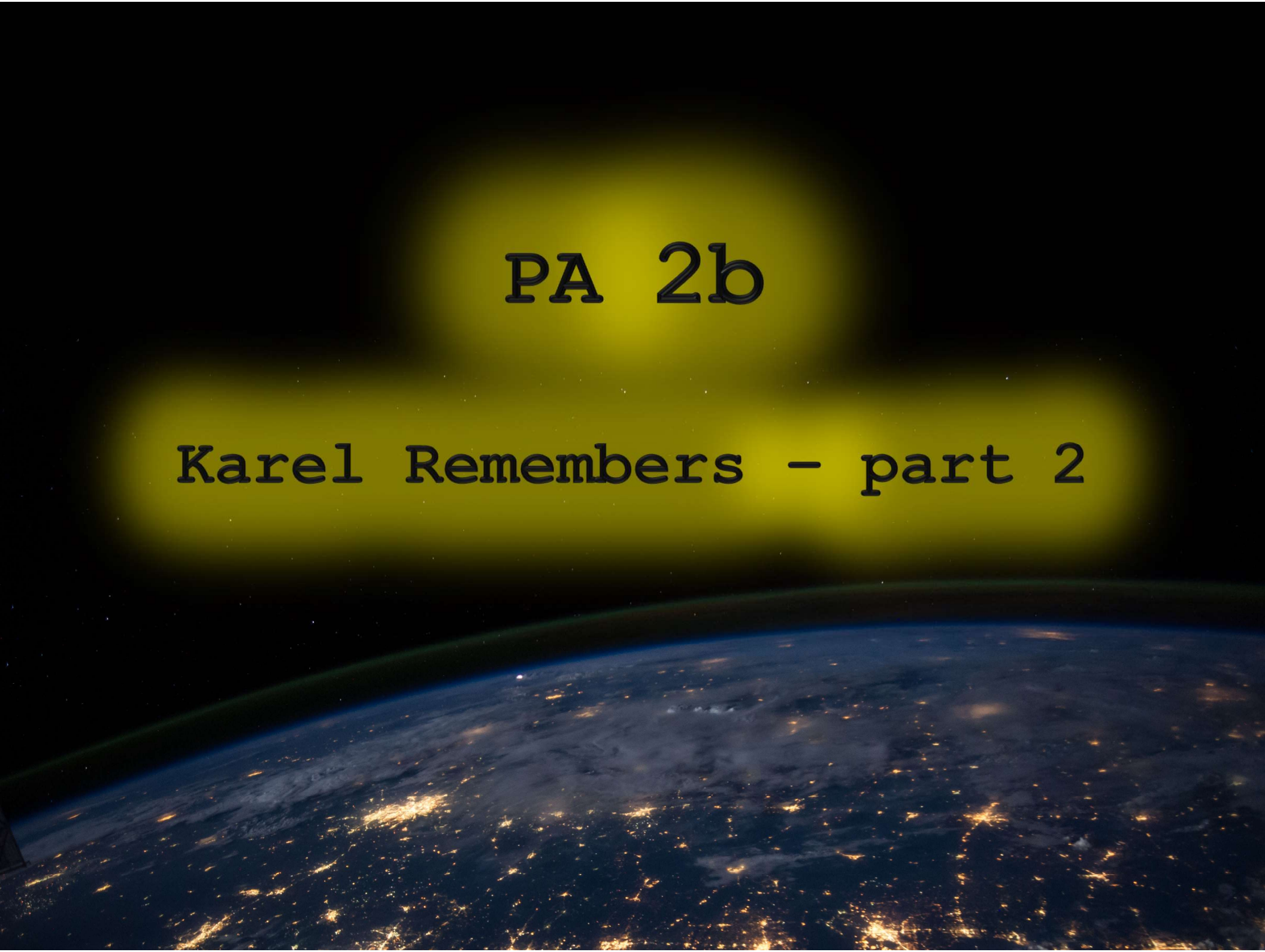

PA 2b
Karel Remembers - part 2

PA 2b

DISCUSSION

- Karel with variables and pointers
  - Repeat no longer works!
  - For some parts, you cannot change the main()!
  - There is no specific “correct” solution

- Advice
  - Only download the starter code once
  - Check that the code compiles before submitting
  - Check messages from submit script
  - Try the other maps we provided (by modifying the settings.json file)
  - Test additional cases (create your own maps)
  - Delete any pause(), say_text(), printf() you are using for debugging

```c
#include <stdio.h>
#define   MAX   3


int main() {
  int x;

  for (x = 0; x < MAX; x++)
       printf("%d/%d ",x, MAX);

}
```

What will be printed?

[A] 0 0 0

[B] 0/3 1/3 2/3

[C] 0/MAX 1/MAX 3/MAX

[D] Something else or the program will enter an infinite loop

[E] It will not compile or result in a run-time error

```c
#include <stdio.h>
#define   MAX   3


int main() {
  int x;

  for (x = 0; x < MAX; x++)
       printf("%d/%d ",x, MAX);

}
```

SOLUTION

What will be printed?

[A] 0 0 0

[B] 0/3 1/3 2/3

[C] 0/MAX 1/MAX 3/MAX

[D] Something else or the program will enter an infinite loop

[E] It will not compile or result in a run-time error

```c
#include <stdio.h>
#define   MAX   3


int main() {
  int x;

  for (x = 0; x < MAX; x++)
        printf("%d/%d ",x, MAX);

}
```

```c
#include <stdio.h>


int main() {
  int x;

  for (x = 0; x < 3; x++)
        printf("%d/%d ",x, 3);

}
```

DISCUSSION

The `#define` specifies a **constant**.

It can be thought off as being resolved as a find-replace before the compilation starts (except inside strings).

It is useful to create meaningfully-named constants that are used throughout the program.

DISCUSSION

```
#include <karel.h>
#define DEBUG 1

int main() {
 karel_setup("settings_01.json");

  int x = 0;
  while ( x < 3 ) {
    if (DEBUG) {
       printf("%d\n",x);
       pause();
    }
    x++;
  }
}
```

The `#define` specifies a **constant**.

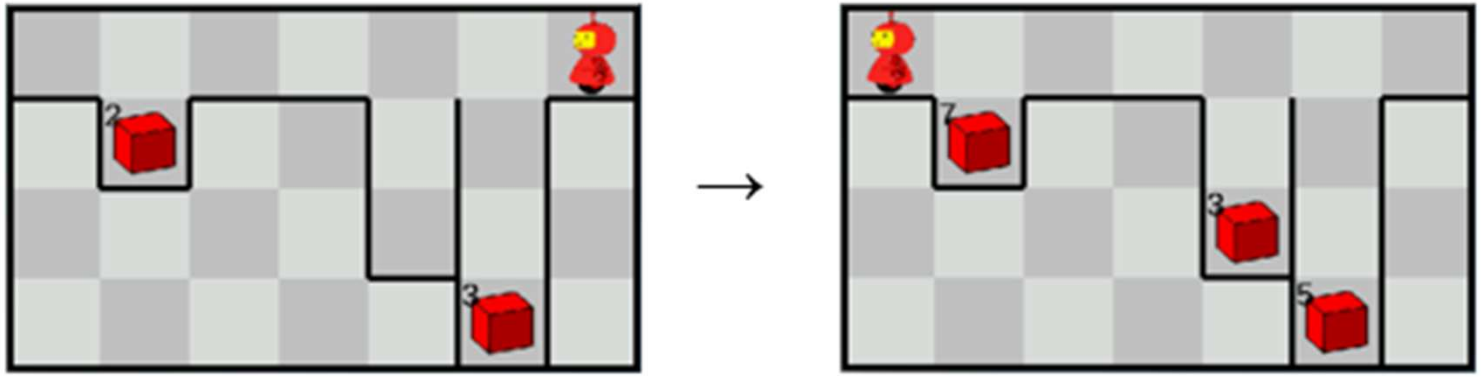
It can be thought off as being resolved as a find-replace before the compilation starts (except inside strings).

It is useful to create meaningfully-named constants that are used throughout the program.

DISCUSSION

PA 2b

`add.c`



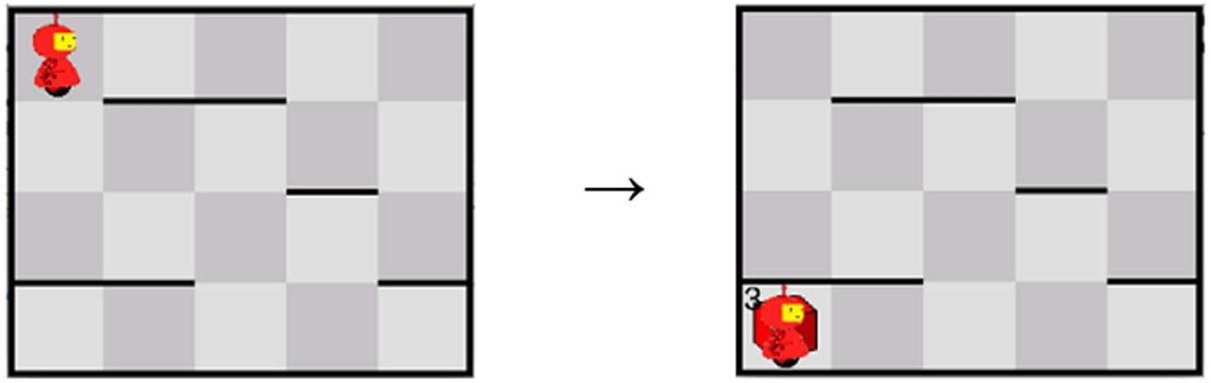
- As part of your code, you need to call our function `decide_items()`
- You should not implement this function yourself

```
void decide_items(int* value)
```

PA 2b

DISCUSSION

`count.c`



- You need to create a function `get_count`
  - Note that this function should not put down any items
- You cannot modify the `main()`
- We will test on a slightly altered version of `main()`

```
int main() {
  karel_setup("settings/settings02_count.json");
  int max, total, i;

  get_count(&max, &total);

  for (i=0;i<total-max;i++)
    put_item();
  turn_off();
}
```

PA 2b

DISCUSSION

`greater.c`

- Not a karel problem

```
number = (base+0) + (base+1) + (base+2) + (base+3) + …
stop adding when number > limit
```

```
E.g., base = 3, limit = 12:
number = 3 + 4 + 5 + 6 = 18
```

- Implement a function that calculates and returns this number

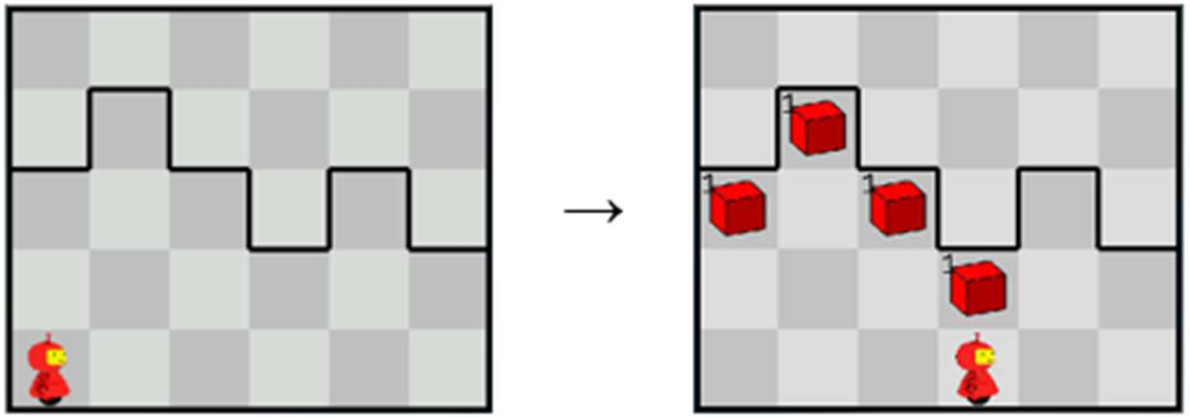
```
int process(int base, int limit)
```

- The goal of his problem is for you to write your own test cases
- We will only test your function and deactivate your `main()`

PA 2b

DISCUSSION

`fixceiling.c`



- You need to create a function `fix_one_column`
  - It should only fix a single column
- You cannot modify the `main()` or use `ITERATIONS` in your function
- We will test on a slightly altered version of `main()`

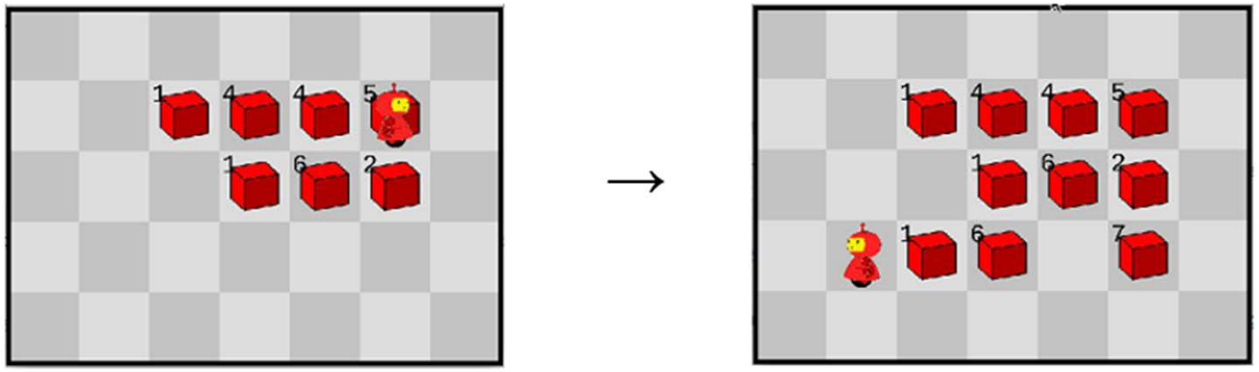
```
#define ITERATIONS 4

int main() {
  karel_setup("settings/settings02_fixceiling.json");
  int num_iterations = ITERATIONS;
  while (1) {
     fix_one_column(num_iterations);
     move();
  }
  turn_off();
}
```

PA 2b

DISCUSSION

`calculate.c`



- You need to create functions `get_digits` and `put_sum_digit`
  - Each loop should put down one digit of the sum
- You cannot modify the `main()`
- We will test on a slightly altered version of `main()`

```
int main() {
  karel_setup("settings/settings02_calculate.json");
  int digit_num1, digit_num2;
  for (;;) {
     get_digits(&digit_num1, &digit_num2);
     put_sum_digit(digit_num1, digit_num2);
  }
  turn_off();
}
```

Important

DISCUSSION

- Plan first

- Test as you go!