Exercise 1:

```
1:  .INCLUDE "M2560DEF.INC"
2:  .ORG 00
3:  LDI R16, 0xFF ; 0. initializing the stack
4:  OUT SPL, R16  ; 0. initializing the stack
5:  LDI R16, 0x21 ; 0. initializing the stack
6:  OUT SPH, R16  ; 0. initializing the stack
7:
8:  LDI R16, 0xFF ; initializing the led-outputth rh
9:  OUT DDRB, R16 ; initializing the led-output
10:
11: main_loop:
12: PUSH R16  ; 1.a Call Setup. A placeholder for the output is allocated
13: LDI R16, 100 ; 1. Call setup (100 is the input value)
14: PUSH R16  ; 1.b CAll Setup. The input value is put on the stack
15: CALL delay_and_invert ; 2. Call Site
16: POP R16 ; 9. poping input values.
17: POP R16 ; 9. Retrieving the output value.
18: OUT PORTB, R16 ; Put the retrieved output on portB (the inverted of what is
    was)
19: JMP main_loop
20:
21: delay_and_invert:
22: PUSH R16 ; 3. saving working registers
23: PUSH R17 ; 3. saving working registers
24: PUSH R26 ; 3. saving working registers
25: PUSH R27 ; 3. saving working registers
26: PUSH R18 ; 3. saving working registers
27:
28: IN R26, SPL ; 4. (Retrienving input values). Setting op the X-register
29: IN R27, SPH ; 4. (Retrienving input values). Setting op the X-register
30: ADIW R26, 10 ;   10 since: 5 from working registers, 3 return adress. 1 input
    value, and 1 extra because the following LD-command pre decrements the X-
    register.
31: LD R18, -X ; 4. (Retrienving input values). Setting op the X-register
32:
33:
34: _10msdelay:  ; 5. Implementing the function body. This loop is used to repeat
    the 10ms delay
35: LDI R17, 160
36: loop2:
37: LDI R16, 199
38: loop1:
39: NOP
40: NOP
41: DEC R16
42: BRNE loop1
43: NOP
44: NOP
45: DEC r17
46: BRNE loop2 ; 10ms delay ended
47:
48: DEC r18
49: BRNE _10msdelay ; 5. The number of time the 10ms delay should be repeated.
50:
51: IN r18, PORTB ; 5. Reading the value of PORTB
52: COM R18              ; 5. Inverting the bits
```

```
53:  ADIW r26, 2 ; 5. updating the x-pointer to point at the right adress. Plussing
     2 (1 for the input which has been decremented and 1 for the output)
54:  ST -X,R18   ; 6. Saving output value 1
55:
56:  POP R18 ; 7. restoring working registers
57:  POP R27 ; 7. restoring working registers
58:  POP R26 ; 7. restoring working registers
59:  POP R17 ; 7. restoring working registers
60:  POP R16 ; 7. restoring working registers
61:  RET
```

Exercise 3 Solution

```
LDI R16, 0xFF ; 0. initializing the stack
OUT SPL, R16  ; 0. initializing the stack
LDI R16, 0x21 ; 0. initializing the stack
OUT SPH, R16  ; 0. initializing the stack


main_loop:
PUSH R16  ; 1.a Call Setup. A placeholder for the output is allocated
LDI R16, 100 ; 1. Call setup (100 is the input value)
PUSH R16  ; 1.b CAll Setup. The input value is put on the stack
CALL add7 ; 2. Call Site
POP R16 ; 9. poping input values.
POP R16 ; 9. Retrieving the output value.
JMP main_loop

add7:
PUSH R16 ; 3. saving working registers
PUSH R17 ; 3. saving working registers
PUSH R26 ; 3. saving working registers
PUSH R27 ; 3. saving working registers

IN R26, SPL ; 4. (Retrienving input values). Setting op the X-register
IN R27, SPH ; 4. (Retrienving input values). Setting op the X-register
ADIW R26, 9 ;   9 since: 4 from working registers, 3 return adress. 1 input value, and
1 extra because the following LD-command pre decrements the X-register.
LD R16, -X ; 4. (Retrienving input values). Setting op the X-register

ldi r17, 7
add r16, r17


ADIW r26, 2 ; 5. updating the x-pointer to point at the right adress. Plussing 2 (1
for the input which has been decremented and 1 for the output)
ST -X,R16   ; 6. Saving output value 1

POP R27 ; 7. restoring working registers
POP R26 ; 7. restoring working registers
POP R17 ; 7. restoring working registers
POP R16 ; 7. restoring working registers
RET
```

Exercise 4 Solution. R16 and R17 contain the previous 2 fabonachi numbers.

```
LDI R16, 0xFF; initializing the stack
```

```
OUT SPL, R16 ; initializing the stack
LDI R16, 0x21; initializing the stack
OUT SPH, R16 ; initializing the stack

ldi r26, 1 ; Original values of the working register (Should be the same after the
function has been executed)
ldi r27, 2
ldi r18, 3
ldi r19, 4

ldi r16, 1
ldi r17, 1

ever:
push r16 ; 1. r16 does not matter. This is for allocating place to the output value
push r16

push r16 ; 1. Call setup
push r17 ; 1. CAll setup

call adderFunc ; 2. Call site
pop r16 ; 9. poping input values.
pop r16 ; 9. poping input values.
pop r16 ; 9. Retrieving output value.
pop r17 ; 9. Retrieving output value.

rjmp ever

adderFunc:
push r26 ; 3. saving working registers
push r27 ; 3. saving working registers
push r18 ; 3. saving working registers
push r19 ; 3. saving working registers

in r26, SPL ; 4. (Retrienving input values). Setting op the X-register
in r27, SPH ;
adiw r26, 10; 4 pushes from working registers, 3 from return adress, 2 inputs, and 1
ekstra.
LD R18, -X  ; 4. retrieving input value r16 = 10
LD R19, -X  ; 4. retrieving input value r17 = 100

add r18, r19 ; 5. implementing the function body
adiw r26, 4  ; 5. updating the x-pointer to point at the right adress.
st -X,r18 ; 6. Saving output value 1
st -X,r19 ; 6. Saving output value 2

pop r19; 7. restoring working registers
pop r18; 7. restoring working registers
pop r27; 7. restoring working registers
pop r26; 7. restoring working registers

ret    ; 8. return from the function
```