İbrahim Burak Tanrıkulu 21827852

Q1-

```
#include <cstdlib.h>
#include <cstdio.h>
#include <signal.h>
#include <time.h>

int time = 0

void startStopwatch(int sig)
{
    time++;
    safe_printf("Stopwatch increased.");
}

void stopStopwatch(int sig)
{
    safe printf("Stopwatch stopped.
                \n Time: %d", time);

    exit(0);
}
int main()
{
   signal(341, &startStopwatch);
   signal(342, &stopStopwatch);
   char input;
   while(1)
   {
      scanf("%c", input);
      if(input == 'A')
         raise(341);
      if(input == 'B')
         raise(342);
   }
}
```

İbrahim Burak Tavukçu 21827852

Q2- We should define a procedure that does computation which we wanted when this interrupt is called. And then, we should jump to actual handler. While we doing these computations, we should use stack to protect actual values of registers.

We cannot change handler, so we change interrupt vector table. Interrupt vector table will hold our function's address and we jump back to actual handler's address.

```
START: ; push all registers to stack.
       movl %eax, (interrupt vector entry) ; We don't know which entry is.
                                           ; Stored actual handler's address.
       movl %esi, OURHANDLER
       movl %edi, interrupt vector entry address
       movl [%edi], %esi      ; Putting our handler's address to IVT.
OURHANDLER:
       ; Do some calculations.
       movl %ecx, %eax        ; ecx has actual handler's address.
       ; Pop all registers from stack except ecx.
       jmp %ecx ; go back to actual handler.
```