Write an interrupt-driven version of the **getsUSART0** function in assembly and C language.

Solution:

The interrupt-driven version of the getsUSART0 subroutine and its test program are as follows:

```
            .include <m2560def.inc>
            .dseg
            .org    0x200
ibuf:       .byte   50
RFlag:      .byte   1
            .cseg
            rjmp    start
            .org    URXC0addr
            rjmp    USART0_RXCIF_ISR
            .org    0xF6
start:      ldi     r20,low(RAMEND)
            out     SPL,r20
            ldi     r20,high(RAMEND)
            out     SPH,r20
            call    initUSART0
            ldi     ZL,low(msg1<<1)     ; output “What is your major?” using polling method
            ldi     ZH,high(msg1<<1)    ;     “
            ldi     r16,1               ;     “
            call    putsUSART0          ;     “
            ldi     XL,low(ibuf)        ; read a string from USART0 using interrupt-driven
method
            ldi     XH,high(ibuf)       ;     “
            call    getsUSART0I         ;     “
            ldi     r16,0x0D            ; move cursor to the start of the next line
            call    putchUSART0         ;     “
            ldi     r16,0x0A            ;     “
            call    putchUSART0         ;     “
            ldi     ZL,low(msg2<<1)     ; output “My major is “ to USART0 using polling method
            ldi     ZH,high(msg2<<1)    ;     “
            ldi     r16,1               ;     “
            call    putsUSART0          ;     “
            ldi     ZL,low(ibuf)        ; output the string read by the getsUSART0I subroutine
            ldi     ZH,high(ibuf)       ;     “
            ldi     r16,0               ;     “
            call    putsUSART0          ;     “
here:       nop
            rjmp    here
msg1:       .db     "What is your major?",0
msg2:       .db     "My major is ",0
```

```
; ------------------------------------------------------------------------------------------------------------------------
----
; This subroutine initializes the USART0 to operate in asynchronous mode with baud rate set to
; 19200. The USART0 is configured to transmit and receive 8-bit data.
; ------------------------------------------------------------------------------------------------------------------------
----
initUSART0:
        ldi     r20,0x06        ; configure RXD/PE0,TXD/PE1 for input, output,
        sts     DDRE,r20        ;     "
        ldi     r20,25          ; set baud rate to 19200 with fOSC = 8 MHz
        sts     UBRR0L,r20      ;     "
        ldi     r20,0           ;     "
        sts     UBRR0H,r20      ;     "
        sts     UCSR0A,r20      ; disable parity, use 16 as divider of transmitter clock
        ldi     r20,0x18        ; enable transmitter, enable receiver, 8-bit data
        sts     UCSR0B,r20      ;     "
        ldi     r20,0x06        ; asynchronous USART, disable parity
        sts     UCSR0C,r20      ;     "
        ret
; ------------------------------------------------------------------------------------------------------------------------
; This function inputs a string from USART0 module using interrupt driven method.
; X register points the data buffer that holds the received string.
; ------------------------------------------------------------------------------------------------------------------------
getsUSART0I:
        lds     r20,UCSR0B      ; enable receive interrupt
        ori     r20,(1<<RXCIE0) ;     "
        sts     UCSR0B,r20      ;     "
        sei
        clr     r20             ; clear receive complete flag
        sts     RFlag,r20       ;     "
wRxLp:  lds     r20,RFlag       ; wait until the whole string is received
        cpi     r20,1           ;     "
        brne    wRxLp           ;     "
        cli                     ; disable interrupt globally
        ret
; ------------------------------------------------------------------------------------------------------------------------
; This is the service routine of the RXCIF0 interrupt.
; ------------------------------------------------------------------------------------------------------------------------
USART0_RXCIF_ISR:
        lds     r22,UDR0        ; read the arrived byte
        cpi     r22,0x0D        ; is the received character a carriage return?
        brne    contR           ; if yes, getsUSART is done
```

```
            ldi     r20,0
            st      X,r20
            lds     r20,UCSR0B          ; disable the RXCIF interrupt
            andi    r20,~(1<<RXCIE0)    ;     "      ;         "
            sts     UCSR0B,r20          ;
            ldi     r20,1               ; set receive complete flag
            sts     RFlag,r20           ;     "
            reti
contR:      st      X,r22               ; save the received character
            mov     r16,r22             ; echo it back to USART0
            call    putchUSART0         ;     "
            cpi     r22,0x08            ; is the received character a backspace?
            brne    noBS
            sbiw    XL,1                ; decrement the input buffer pointer
            ldi     r16,0x20            ; output a space character
            call    putchUSART0         ;     "
            ldi     r16,0x08            ; output a backspace character
            call    putchUSART0
            reti
noBS:       adiw    XL,1                ; increment the input buffer
            reti
; ---------------------------------------------------------------------------------------------------------------
; The following subroutine outputs the character passed in r16 to MEGA device USART0
; using the polling method. The character is less than 9 bits.
; ---------------------------------------------------------------------------------------------------------------
putchUSART0:
            lds     r20,UCSR0A          ; make sure data register is empty before
            sbrs    r20,UDRE0           ; outputting the character
            rjmp    putchUSART0         ;     "
            sts     UDR0,r16            ; output the character (less than 9 bits)
            ret
; ---------------------------------------------------------------------------------------------------------------
; The following subroutine outputs a string pointed to by Z to USART0. The string is stored in
; the program memory or data memory. r16 indicates whether the string is in program memory
(=1)
; or data memory (=0).
; ---------------------------------------------------------------------------------------------------------------
putsUSART0:
            cpi     r16,1               ; is string in program memory?
            breq    pstr                ;     "
dstr:       ld      r16,z+              ; string is in data memory
            cpi     r16,0
```

```
            breq    done            ; reach the end of string?
            rcall   putchUSART0     ; output the next character
            rjmp    dstr
pstr:       lpm     r16,z+          ; string is in program memory
            cpi     r16,0
            breq    done            ; reach the end of string?
            rcall   putchUSART0     ; output the next character
            rjmp    pstr
done:       ret
```

The C language version of the getsUSART0I function and its test program are as follows:

```
#include <avr\io.h>
#include <avr\interrupt.h>
void initUSART0(void);
void putchUSART0(unsigned char cx);
void putsUSART0(unsigned char *ptr);
void getsUSART0I(void);
void newline(void);
unsigned char *msg1  = "What is your major?";
unsigned char *msg2  = "My major is ";
unsigned char msg[4] = {0x0D,0x0A,0,0};
unsigned char *ptr;             // pointer to a string
volatile unsigned char RXComplete;      // RXComplete must be declared as volatile!!
unsigned char tmp, ibuf[32]; // input buffer
void main(void)
{
    initUSART0();
    newline();
    putsUSART0(msg1);
    ptr = &ibuf[0]; // set up input buffer pointer before input
    getsUSART0I();
    newline();
    putsUSART0(msg2);
    putsUSART0(&ibuf[0]);
    while(1);
}
void initUSART0(void)
{
    DDRE        = 0x02; // configure pins RXD0/PE0, TXD0/PE1 for input and output
    UBRR0       = 25;  // configure USART0 to shift at 19200 Hz with 8-MHz CPU clock
    UCSR0A      = 0;    // disable multiprocessor communication, use 16 as divider of transmitter
clock
```

```
    UCSR0B      = 0x18; // enable transmitter, enable receiver, send 8-bit data
    UCSR0C      = 0x06; // select asynchronous USART, disable parity, use one-stop bit
}
// ---------------------------------------------------------------------------------------------------------
// This function reads a string from the USART0 using the interrupt-driven method.
// ---------------------------------------------------------------------------------------------------------
void getsUSART0I(void)
{
    RXComplete = 0;
    UCSR0B |= (1 << RXCIE0); // enable RXC0 flag interrupt
    sei();
    while(RXComplete == 0); // wait until the reception is completed
    cli();
}
// ---------------------------------------------------------------------------------------------------------
// This is the service routine for the reception complete interrupt.
// ---------------------------------------------------------------------------------------------------------
ISR(USART0_RX_vect)
{
    tmp = UDR0;
    if(tmp == 0x0D){
        *ptr = 0; // terminate the string with a NULL character
        RXComplete = 1;
        UCSR0B &= ~(1 << RXCIE0); // disable reception complete interrupt
    }
    else {
        *ptr = tmp; // save received character in ibuf
        putchUSART0(tmp); // echo back to USART0
        if(tmp == 0x08){
            putchUSART0(0x20);
            putchUSART0(0x08);
            ptr--; // decrement the ibuf pointer
        }
        else
            ptr++;
    }
}
// ---------------------------------------------------------------------------------------------------------
// The following function outputs a character to the USART0 module using the polling method.
// ---------------------------------------------------------------------------------------------------------
void putchUSART0 (unsigned char cx)
{
```

```
    while(!(UCSR0A & (1 << UDRE0))); // wait for empty transmit buffer
    UDR0 = cx;
}
// -------------------------------------------------------------------------------------------------
// This function moves the cursor to the beginning of the next line.
// -------------------------------------------------------------------------------------------------
void newline(void)
{
    putchUSART0(0x0D);
    putchUSART0(0x0A);
}
// -------------------------------------------------------------------------------------------------
// This function outputs a string to the USART0 by calling the putchUSART0 continuously
// until the whole string has been sent out.
// -------------------------------------------------------------------------------------------------
void putsUSART0(unsigned char *ptr)
{
    while(*ptr){
        putchUSART0(*ptr);
        ptr++;
    }
}
```