# Exercise 7 Answers

Assume that the following statements have already been defined for all problems.

PORTB EQU $0001

DDRB EQU $0003

PTP EQU $0258

DDRP RQU $025A

PTH EQU $0260

DDRH EQU $0262

PPSH EQU $0265

PIEH EQU $0266

PIFH EQU $0267

1. 1. MOVB #%00001111,DDRH
   2. MOVB #%11111111,DDRB
   3. BSET DDRB,%00000011
   4. BCLR DDRP,%00011100
2. The answers below are not unique.
   1. MOVB #%00011100, PORTB
   2. BSET PORTB, %00011100
   3. MOVB #%10110101, PTP
   4. BSET PTP,%00111100
   5. BRSET PTH,#%00001000,skip

BSET PTP,#%00001000

skip …

* 1. BRCLR PTH,%00001100,skip1

BRA skip2

skip1 BSET PTP,%00001111

skip2 …

* 1. BRSET PTH,%00001100,skip

BSET PTP,%00001111

skip …

1. The answers below are not unique
   1. uc\_to\_ul ldaa #0 ; extend B to D as unsigned value

tfr D,X ;

ldd #0 ; 0’s to upper two bytes of answer

rts

unsigned long uc\_to\_ul(signed char); // C function prototype

outlong = uc\_to\_ul(inchar); // example C function call

* 1. c\_to\_l ldy #0 ; assume positive char

clra

tstb ; char passed in B, get sign

bpl done ; positive assumption correct

ldaa #$FF ; correct to negative

ldy #$FFFF

done tfr D,X ; rearrange answer to D:X for

tfr Y,D ; CodeWarrior convention

rts

signed long uc\_to\_ul(signed char); // C function prototype

outlong = uc\_to\_ul(inchar); // example C function call

subtwo tfr D,Y ; points to overflow

clr 0,Y

ldx 2,SP ; points to result

ldd 10,SP ; least two bytes of minuend

subd 6,SP ; sub least two bytes of subtrahend

std 2,X ; save least two bytes of difference

ldd 8,SP ; most sig. two bytes of minuend

sbcb 5,SP ; sub most sig. two bytes tracking

sbca 4,SP ; carry (borrow) bits

bvc skip ; indicate signed overflow if it occurs

movb #1,0,Y

skip std 0,X ; store most sig two bytes of difference

rts

// C function prototype

signed long subtwo(signed long, signed long, signed long\*, unsigned char);

// example Cfunction call

subtwo(minuend, subtrahend, \*difference, overflow);



addlist tfr D,Y ; move length to Y so D can be used for addition

ldd #0 ; set running sum to 0

ldx 2,SP ; point to list

cpy #0

loop beq endsub ; while more items are present…

addd 0,X ; add next two-byte value

leax 2,X ; point to next

dey ; decrement number of items remaining

bra loop

endsub rts ; return with running sum in D

unsigned int addlist(int\*, int); //function prototype

sum = addlsit(&array[0], arraylength); //function call

or, since the name of an array without [] implies the address of the first element

sum = addlist(array, arraylength);