```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.

;-------------------------------------------------------------------------------
            .data                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
            .retainrefs                     ; And retain any sections that have

;You can uncomment this portion and use this array to test your code
;Do not submit your results for this array !!
LENGTH: .set 16
array:  .word 4, -3, -7, 6, -5, -1, 2, 0

;LENGTH:    .set 512
;array:     .space LENGTH
min_value:  .word 32767
;-------------------------------------------------------------------------------
            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.

;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer

;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------

            mov.w   #array, R8              ; sort takes the address of
            mov.w   #LENGTH/2, R10

            call    #sort

done:       jmp     done
            nop


;-------------------------------------------------------------------------------
; Subroutine: sort
; Inputs: R8 pointer to word array -- returned unchanged
;           R8 contains the 16-bit starting address of an array
;               The array contains N 16-bit signed integers
;
;         R10 = N, the number of elements in the array -- returned unchanged
;
; Output: The subroutine sorts the elements in a given array from smallest to largest
```

```
58 ;           You will implement selection sort
59 ;
60 ; All core registers in R4-R15 unchanged
61 ; Subroutine does not access any global variables or defined constants
62 ;-------------------------------------------------------------------------------
63 sort:
64             push.w  R7                      ; R7 is array position counter
65             push.w  R9                      ; R9 is smallest num pointer
66             mov.w   #0, R7
67
68 sort_loop:
69             call    #select                 ; Select next lowest number with R9
70
71             add.w   R7, R8                  ; Add index to array address
72             call    #swap                   ; Swap R8 and R9
73
74             incd.w  R7
75
76             cmp.w   R10, R7
77             jne     sort_loop               ; loop if R7 < R10
78
79
80             pop.w   R9                      ; restore and return
81             pop.w   R7
82             ret
83
84 ;-------------------------------------------------------------------------------
85 ; Subroutine: select
86 ; Inputs: R8 pointer to word array -- returned unchanged
87 ;           R8 contains the 16-bit starting address of an array
88 ;               The array contains N 16-bit signed integers
89 ;
90 ;         R10 = N, the number of elements in the array -- returned unchanged
91 ;
92 ; Output: R9 pointer to the smallest element in the array
93 ;           R9 contains the 16-bit address of the smallest element
94 ;
95 ; All other core registers in R4-R15 unchanged
96 ; Subroutine does not access any global variables or defined constants
97 ;-------------------------------------------------------------------------------
98 select:
99             mov.w   #0x7FFF, R9             ; init min_value = max_value
100            push.w  R5                      ; R5 is index
101            clr.w   R5
102            rla.w   R10
103
104
105 compare_to_min:
106            cmp.w   #R9, array(R5)          ; compare current element to max
107            jge     next_element            ; if larger than min, proceed to next
108
109            add.w   #array, R5
110            mov.w   R5, R9                  ; update min
111            sub.w   #array, R5
112
113
114 next_element:
```

```
115             incd.w  R5                      ; increment counter
116             cmp.w   R10, R5
117             jl      compare_to_min          ; loop if counter == length
118
119
120             rra.w   R10
121             pop.w   R5                      ; Restore R5
122             ret
123
124 ;-------------------------------------------------------------------------------
125 ; Subroutine: swap
126 ; Inputs: R8 address of word with value x -- returned unchanged
127 ;         R9 address of word with value y -- returned unchanged
128 ;
129 ; Output: The subroutine swaps x and y
130 ;           i.e., the word with address in R8 will have value y
131 ;                 the word with address in R9 will have value x
132 ;
133 ; All core registers in R4-R15 unchanged
134 ; Subroutine does not access any global variables or defined constants
135 ; It is allowed to use the stack
136 ;-------------------------------------------------------------------------------
137 swap:
138
139         push.w  R8
140         push.w  R9
141
142         pop.w   R8
143         pop.w   R9
144
145         ret
146
147 ;-------------------------------------------------------------------------------
148 ; Stack Pointer definition
149 ;-------------------------------------------------------------------------------
150             .global __STACK_END
151             .sect   .stack
152
153 ;-------------------------------------------------------------------------------
154 ; Interrupt Vectors
155 ;-------------------------------------------------------------------------------
156             .sect   ".reset"                ; MSP430 RESET Vector
157             .short  RESET
158
159
```