```
1:  # Program to sort an array of values in ascending order, using Selection Sort.
2:  # Uses functions "findSmallest" and "swap" repeatedly as needed,
3:  # uses function printList at the beginning and end of program to display list values.
4:  # array "values" and length of array "valueCount" are given.
5:  # 3 display strings are also added for the printList function.
6:
7:
8:  # Written by Kollen G
9:
10:
11:         .data
12:         .align  2
13: displayPre: .asciiz "\nBeginning list: "
14: displayPost:    .asciiz "\nSorted list: "
15: comma:  .asciiz ", "
16: valueCount: .word   10
17: values: .word   42, 66, 613, -29, 57, 212, 87, 2, -86, 9
18:
19: #-------------------------------
20:         .text
21:         .globl  main
22:
23: main:
24:     la  $s3, values # s3 : address of array
25:         lw  $s0, valueCount # s0 : cnt
26:         move    $s1, $0     # s1 : outer index "i"
27:         move    $s2, $0     # s2 : index of min value
28:
29:         # setup printList func call
30:         move    $a1, $s0
31:     move    $a2, $s3
32:     la  $a0, displayPre #load display string
33:     li  $v0, 4      #code to print string
34:     syscall
35:     jal printList
36:
37: loopO:  bge $s1, $s0, brk   # while (i < cnt) "outer loop"
38:     move    $s2, $s1    # s2: index of min value = s1: outer index "i"
39:
40:     #setup findSmallest func call
41:     move    $a0, $s1
42:     move    $a1, $s0
43:     move    $a2, $s3
44:     move    $a3, $s2
45:     jal findSmallest
46:     move    $s2, $v0
47:
48:     beq $s1, $s2, cont  # if (i != index of min)
49:     # setup swap func call
50:     move    $a0, $s1
51:     move    $a1, $s2
52:     move    $a2, $s3
53:     jal swap
54:
55: cont:   addi    $s1,$s1,1   # increment i++
```

```
56:     j loopO
57:
58: brk:    # setup printList func call
59:     move    $a1, $s0
60:     move    $a2, $s3
61:     la  $a0, displayPost#load display string
62:     li  $v0, 4      #code to print string
63:     syscall
64:     jal printList
65:     j exit
66:
67: #----------------- Exit ---------------------
68: exit:   li  $v0, 10
69:     syscall
70:
71:
72: #*****************************************************************
73:     # printList function
74:     #
75:     # a0 - outer index "i"
76:     # a1 - cnt
77:     # a2 - address of array
78:     #
79: printList:
80: #--------------- Usual stuff at function beginning  ---------------
81:         addi    $sp, $sp, -24   # allocate stack space for 6 values
82:         sw  $ra, 20($sp)    # store off the return addr, etc
83:         sw  $s0, 16($sp)
84:         sw  $s1, 12($sp)
85:         sw  $s2, 8($sp)
86:         sw  $s3, 4($sp)
87:         sw  $s4, 0($sp)
88: #------------------------- function body -------------------------
89:     move    $s0, $0     # s0 : n
90:     move    $s1, $a1    # s1 : cnt
91:     move    $s3, $a2    # s3 : address of array
92:
93: loopF:  bge $s0, $s1, d # while (n < cnt)
94:
95:     sll     $s4, $s0, 2 # s4: offset 4 for array[n] (4*n)
96:         add     $s4, $s4, $s3   # s4: addr of array[n]
97:         lw      $t1, 0($s4) # t1 <-- array[n] value
98:
99:     li  $v0, 1      #code to print int
100:    move    $a0, $t1    #load array[n] value
101:    syscall         #print
102:    la  $a0, comma  #load display string
103:    li  $v0, 4      #code to print string
104:    syscall
105:
106:    addi    $s0,$s0,1   #increment n++
107:    j loopF
108:
109: d: move    $v0, $s0
110: #-------------------- Usual stuff at function end -----------------
```

```
111:         lw     $ra, 20($sp)    # restore the return address and
112:         lw $s0, 16($sp)    # other used registers...
113:         lw $s1, 12($sp)
114:         lw $s2, 8($sp)
115:         lw $s3, 4($sp)
116:         lw $s4, 0($sp)
117:         addi   $sp, $sp, 24
118:         jr      $ra
119:
120:
121: #*****************************************************************
122:    # findSmallest function
123:    #
124:    # a0 - outer index "i"
125:    # a1 - cnt
126:        # a2 - address of array
127:        # a3 - index of min
128:        #
129:        # v0 - index of min value
130: findSmallest:
131: #--------------- Usual stuff at function beginning  ---------------
132:         addi    $sp, $sp, -24  # allocate stack space for 6 values
133:         sw $ra, 20($sp)    # store off the return addr, etc
134:         sw $s0, 16($sp)
135:         sw $s1, 12($sp)
136:         sw $s2, 8($sp)
137:         sw $s3, 4($sp)
138:         sw $s4, 0($sp)
139: #------------------------- function body -------------------------
140:    move    $s0, $a0    # s0 : outer index "i"
141:    addi    $s1, $s0, 1 # s1 : inner index "j" = i + 1
142:    move    $s2, $a1    # s2 : cnt
143:    move    $s3, $a2    # s3 : address of array
144:    move    $s4, $a3    # s4 : min index
145:
146: loopI: bge $s1, $s2, brk2  # while (j < cnt) "inner loop"
147:
148:    sll     $s5, $s1, 2     # s5: offset 4 for array[j]
149:         add     $s5, $s5, $s3   # s5: address of array[j]
150:    lw  $t1, 0($s5) # t1 : value array[j]
151:
152:    sll $s6, $s4, 2 # s6 : offset 4 for array[min]
153:    add $s6, $s6, $s3   # s6 : address of array[min]
154:    lw  $t2, 0($s6) # t2 : value array[min]
155:
156:    bge $t1, $t2, cont2 # if (array[j] < array[min])
157:    move    $s4, $s1    # s4 : min index = j
158: cont2: addi    $s1, $s1, 1 # increment j++
159:    j loopI
160:
161: brk2:  move    $v0, $s4
162:
163: #------------------- Usual stuff at function end -----------------
164:         lw     $ra, 20($sp)    # restore the return address and
165:         lw $s0, 16($sp)    # other used registers...
```

```
166:         lw $s1, 12($sp)
167:         lw $s2, 8($sp)
168:         lw $s3, 4($sp)
169:         lw $s4, 0($sp)
170:         addi   $sp, $sp, 24
171:         jr      $ra
172:
173:
174: #*****************************************************************
175:    # swap function
176:    #
177:    # a0 - outer index "i"
178:    # a1 - index of min value
179:        # a2 - address of array
180:        #
181: swap:
182: #--------------- Usual stuff at function beginning  ---------------
183:         addi    $sp, $sp, -24  # allocate stack space for 6 values
184:         sw $ra, 20($sp)    # store off the return addr, etc
185:         sw $s0, 16($sp)
186:         sw $s1, 12($sp)
187:         sw $s2, 8($sp)
188:         sw $s3, 4($sp)
189:         sw $s4, 0($sp)
190: #-------------------------- function body -------------------------
191:        sll     $s0, $a0, 2     # s0 : offset for array[i] of 4
192:         add     $s0, $s0, $a2   # s0 : address of array[i]
193:         lw      $s1, 0($s0)     # s1 : value array[i]
194:
195:         sll     $s2, $a1, 2     # s2 : offset for array[min] of 4
196:         add     $s2, $s2, $a2   # s2 : address of array[min]
197:         lw      $s3, 0($s2)     # s3 : value array[min]
198:
199:         sw      $s1, 0($s2)     # array[min] = array[i]
200:         sw      $s3, 0($s0)     # array[i] = array[min]
201:
202:
203: #-------------------- Usual stuff at function end -----------------
204:         lw     $ra, 20($sp)    # restore the return address and
205:         lw $s0, 16($sp)    # other used registers...
206:         lw $s1, 12($sp)
207:         lw $s2, 8($sp)
208:         lw $s3, 4($sp)
209:         lw $s4, 0($sp)
210:         addi   $sp, $sp, 24
211:         jr      $ra
212:
```