```
-----------------------------------------------------------------------------
--
      Object
Addr  code   Symbol   Mnemon  Operand     Comment
-----------------------------------------------------------------------------
--
             ;; VectorManipulation.pep
             ;;
             ;; AUCSC 250
             ;; OCT 30, 2018
             ;; Philippe Nadon
             ;;
             ;; A simple program for manipulating vectors
             ;; Starting method is main


             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; METHODS HEADER
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;;
             ;; void main():
             ;;       Entry method, runs program
             ;;
             ;; void inVect ( int size, int[] vector):
             ;;       Obtains vector contents from user
             ;;
             ;; void prinVect ( int size, int[] vector):
             ;;     Prints the vector
             ;;
             ;; void rotLeft ( int size, int[] vector):
             ;;     Shifts the vector's cells left
             ;;
             ;; void rotRight ( int size, int[] vector):
             ;;     Shifts the vector's cells right
             ;;
             ;; boolean exchange ( int loc1, int loc2,
             ;;           int size, int[] vector):
             ;;     Swaps vector[ loc1] and vector[ loc2]
             ;;
             ;; boolean chckInpt (int size), A = loc1, X = loc2:
             ;;     Ensures loc1 and loc2 are valid
             ;;
             ;; int malloc (), A = size:
             ;;     Moves the heap pointer to allocate
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

             true:    .EQUATE 1
             false:   .EQUATE 0

0000  240004          CALL    main
0003  00              STOP

             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; void main ()
             ;;
             ;; Prompts for inputs and runs corresponding
             ;; methods
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             size:    .EQUATE 0           ;local var #2d
             vector:  .EQUATE 2           ;local var #2h
             inptSize:.EQUATE -4          ; input param #2d
             inptVect:.EQUATE -2          ; input param #2d
0004  580004 main:    SUBSP   4,i         ;push #vector #size
```

```
0007  490212           STRO    sizeMsg,d
000A  330000           DECI    size,s
000D  C30000           LDWA    size,s
0010  A00000           CPWA    0,i
0013  1E0019           BRGT    allocVec
0016  C00001           LDWA    1,i

0019  0A     allocVec:ASLA
001A  E30000           STWA    size,s
001D  240208           CALL    malloc
0020  EB0002           STWX    vector,s

             ; call inVect
0023  C30000           LDWA    size,s
0026  E3FFFC           STWA    inptSize,s  ; -4 on SP

0029  C30002           LDWA    vector,s
002C  E3FFFE           STWA    inptVect,s  ; -2 on SP

002F  580004           SUBSP   4,i         ; push #vector #size
0032  2400FA           CALL    inVect
0035  500004           ADDSP   4,i         ;pop #vector #size

             ;; call to prinVect
0038  C30000 mLoop:    LDWA    size,s
003B  E3FFFC           STWA    inptSize,s  ; -4 on SP

003E  C30002           LDWA    vector,s
0041  E3FFFE           STWA    inptVect,s  ; -2 on SP

0044  580004           SUBSP   4,i         ; push #vector #size
0047  240124           CALL    prinVect
004A  500004           ADDSP   4,i         ;pop #vector #size

004D  49022B           STRO    cmdPrmpt,d

             ; Evaluate input and branch
0050  D9FC15           LDBX    charIn,d    ;Extra char in charIn
0053  780045           SUBX    'E',i
0056  160063           BRLT    caseErr
0059  A8002D           CPWX    45,i
005C  1E0063           BRGT    caseErr
005F  0B               ASLX
0060  1302E4           BR      choiceJT,x

             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ; Default case
0063  490293 caseErr:  STRO    errInput,d
0066  D9FC15           LDBX    charIn,d    ;Extra char in charIn
0069  120038           BR      mLoop

             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; Case E / e: call exchange
             inptLoc1:.EQUATE -8           ; input param for exchange #2h
             inptLoc2:.EQUATE -6           ; input param for exchange #2h
006C  4902B2 caseE:    STRO    xchngMsg,d
006F  33FFF8           DECI    inptLoc1,s
0072  C3FFF8           LDWA    inptLoc1,s
0075  63FFF8           ADDA    inptLoc1,s  ; double the index
0078  E3FFF8           STWA    inptLoc1,s

007B  33FFFA           DECI    inptLoc2,s
007E  C3FFFA           LDWA    inptLoc2,s
0081  63FFFA           ADDA    inptLoc2,s  ; double the index
```

```
0084  E3FFFA          STWA    inptLoc2,s

0087  C30000          LDWA    size,s
008A  E3FFFC          STWA    inptSize,s  ; -4 on SP

008D  C30002          LDWA    vector,s
0090  E3FFFE          STWA    inptVect,s  ; -2 on SP

0093  580008          SUBSP   8,i         ; push #exLoc2 #exLoc1 #vector 
#size
0096  240140          CALL    exchange
0099  500008          ADDSP   8,i         ;pop #exLoc2 #exLoc1 #vector #size

             ; check if input was valid
009C  A80000          CPWX    false,i
009F  1E00A5          BRGT    CaseEEnd

00A2  490267          STRO    errExMsg,d

00A5  120038 CaseEEnd:BR      mLoop

             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; Case L / l: call rotLeft
00A8  C30000 caseL:   LDWA    size,s
00AB  E3FFFC          STWA    inptSize,s  ; -4 on SP

00AE  C30002          LDWA    vector,s
00B1  E3FFFE          STWA    inptVect,s  ; -2 on SP

00B4  580004          SUBSP   4,i         ; push #vector #size
00B7  2401A9          CALL    rotLeft
00BA  500004          ADDSP   4,i         ;pop #vector #size

00BD  D9FC15          LDBX    charIn,d    ;Extra char in charIn
00C0  120038          BR      mLoop

             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ; Case Q / q: print quit message and return from main
00C3  4902D0 caseQ:   STRO    exitMsg,d

00C6  C30000          LDWA    size,s
00C9  E3FFFC          STWA    inptSize,s  ; -4 on SP

00CC  C30002          LDWA    vector,s
00CF  E3FFFE          STWA    inptVect,s  ; -2 on SP

00D2  580004          SUBSP   4,i         ; push #vector #size
00D5  240124          CALL    prinVect
00D8  500004          ADDSP   4,i         ;pop #vector #size

00DB  500004          ADDSP   4,i         ;pop #vector #size
00DE  01              RET                 ;main

             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; Case R / r: call rotRight
00DF  C30000 caseR:   LDWA    size,s
00E2  E3FFFC          STWA    inptSize,s  ; -4 on SP

00E5  C30002          LDWA    vector,s
00E8  E3FFFE          STWA    inptVect,s  ; -2 on SP

00EB  580004          SUBSP   4,i         ; push #vector #size
00EE  2401D7          CALL    rotRight
00F1  500004          ADDSP   4,i         ;pop #vector #size
```

```
00F4  D9FC15          LDBX    charIn,d    ;Extra char in charIn
00F7  120038          BR      mLoop



             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; void inVect (int size, int[] vector)
             ;;
             ;; Takes inputs size and int[] vector, and
             ;; fills each cell of the vector until the end
             ;; is reached
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             index:   .EQUATE 0           ; local parameter #2d
             inSize:  .EQUATE 4           ;formal parameter #2d
             inVector:.EQUATE 6           ;formal parameter #2h
00FA  580002 inVect:  SUBSP   2,i         ; push #index
00FD  C80000          LDWX    0,i
0100  0D     inVLoop: ASRX
0101  680001          ADDX    1,i

0104  4902D9          STRO    inVectP1,d
0107  EB0000          STWX    index,s
010A  3B0000          DECO    index,s
010D  4902DC          STRO    inVectP2,d

0110  780001          SUBX    1,i
0113  0B              ASLX
0114  370006          DECI    inVector,sfx
0117  680002          ADDX    2,i
011A  AB0004          CPWX    inSize,s
011D  160100          BRLT    inVLoop
0120  500002          ADDSP   2,i         ; pop #index
0123  01              RET                 ;inVect



             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; void prinVect (int size, int[] vector)
             ;;
             ;; Takes inputs size and int[] vector, and
             ;; prints each cell of the vector until the end
             ;; is reached
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;index: .EQUATE 0; local parameter
             ;inSize: .EQUATE 4 ;formal parameter
             ;inVector: .EQUATE 6 ;formal parameter
0124  580002 prinVect:SUBSP   2,i         ; push #index
0127  C80000          LDWX    0,i
012A  4902E2          STRO    newLine,d
012D  3F0006 prinLoop:DECO    inVector,sfx

0130  4902E0          STRO    spcIsSpc,d

0133  680002          ADDX    2,i
0136  AB0004          CPWX    inSize,s
0139  16012D          BRLT    prinLoop

013C  500002          ADDSP   2,i         ; pop #index
013F  01              RET                 ;prinVect



             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
            ;; boolean exchange (int size, int[] vector,
            ;;              int exLoc1, int exLoc2)
            ;;
            ;; Takes inputs size, int[] vector, int exLoc1,
            ;; and int exLoc2, and swaps the two cells in
            ;; vector defined by the values of exLoc1 and
            ;; exLoc2, which represent indices
            ;;
            ;; Returns false via index register if exLoc1 or
            ;; exLoc2 were invalid, true otherwise
            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            chInSize:.EQUATE -2          ; chckInput param #2d
            exVal:   .EQUATE 0           ; local var #2d
            exLoc1:  .EQUATE 4           ; formal param #2d
            exLoc2:  .EQUATE 6           ; formal param #2d
            exSize:  .EQUATE 8           ; formal param #2d
            exVect:  .EQUATE 10          ; formal param #2h
0140  580002 exchange:SUBSP   2,i        ;push #exVal

            ; Call chckInput (int chckSize), A = exLoc1, X = exLoc2
0143  C30008          LDWA    exSize,s
0146  E3FFFE          STWA    chInSize,s

0149  C30004          LDWA    exLoc1,s
014C  CB0006          LDWX    exLoc2,s

014F  580002          SUBSP   2,i        ;push #exSize
0152  240183          CALL    chckInpt
0155  500002          ADDSP   2,i        ; pop #exSize

0158  A80000          CPWX    0,i
015B  14017F          BRLE    xchngEnd

            ; Store exVect[ exLoc1] in exVal
015E  CB0004          LDWX    exLoc1,s
0161  C7000A          LDWA    exVect,sfx
0164  E30000          STWA    exVal,s

            ; Store exVect[ exLoc2] in exVect[ exLoc1]
0167  CB0006          LDWX    exLoc2,s
016A  C7000A          LDWA    exVect,sfx
016D  CB0004          LDWX    exLoc1,s
0170  E7000A          STWA    exVect,sfx

            ; Store exVal in exVect[ exLoc2]
0173  CB0006          LDWX    exLoc2,s
0176  C30000          LDWA    exVal,s
0179  E7000A          STWA    exVect,sfx

            ; X = 1 if valiud input, 0 otherwise
017C  C80001          LDWX    true,i
017F  500002 xchngEnd:ADDSP   2,i        ;pop #tempVal
0182  01              RET                ;exchange



            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            ;; boolean chckInpt (int size)
            ;;
            ;; Takes int size via a parameter and int exLoc1
            ;; and int exLoc2 via accumulator and index
            ;; register respectively
            ;;
            ;; Checks to see if exLoc1 and exLoc2 refer
```

```
             ;; to valid indices for a vector with a length
             ;; of size
             ;;
             ;; Returns 0 via index register if exLoc1 or exLoc2
             ;; were invalid, 1 otherwise
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             chckSize:.EQUATE 2            ; formal param #2d
0183  A80000 chckInpt:CPWX    0,i
0186  1601A5          BRLT    badChck
0189  680002          ADDX    2,i
018C  AB0002          CPWX    chckSize,s
018F  1E01A5          BRGT    badChck

0192  A00000          CPWA    0,i
0195  1601A5          BRLT    badChck
0198  600002          ADDA    2,i
019B  A30002          CPWA    chckSize,s
019E  1E01A5          BRGT    badChck

             ; no invalid input checks triggered
01A1  C80001          LDWX    true,i
01A4  01              RET                  ;checkInpt

             ; invalid input check triggered
01A5  C80000 badChck: LDWX    false,i
01A8  01              RET                  ;chckInput



             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; void rotLeft (int size, int[] vector)
             ;;
             ;; Takes inputs size and int[] vector, and
             ;; sequentially replaces the next cell in
             ;; vector with the previous cell's content
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;index: .EQUATE 0
             tempVal: .EQUATE 2            ; local param #2d
             rotSize: .EQUATE 6            ; formal param #2d
             rotVect: .EQUATE 8            ; formal param #2h
01A9  580004 rotLeft: SUBSP   4,i          ; push #vector #size
01AC  C40008          LDWA    rotVect,sf
01AF  E30002          STWA    tempVal,s
01B2  C80002          LDWX    2,i

01B5  AB0006 rotLLoop:CPWX    rotSize,s
01B8  1C01CA          BRGE    rotLEnd

01BB  C70008          LDWA    rotVect,sfx
01BE  780002          SUBX    2,i
01C1  E70008          STWA    rotVect,sfx
01C4  680004          ADDX    4,i
01C7  1201B5          BR      rotLLoop

01CA  780002 rotLEnd: SUBX    2,i
01CD  C30002          LDWA    tempVal,s
01D0  E70008          STWA    rotVect,sfx
01D3  500004          ADDSP   4,i          ; pop #vector #size
01D6  01              RET                  ;rotLeft


             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; void rotRight (int size, int[] vector)
             ;;
             ;; Takes inputs size and int[] vector, and
             ;; sequentially replaces the previous cell
             ;; in vector with the next cell's content
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;index: .EQUATE 0
             ;tempVal: .EQUATE 2 ; local param
             ;rotSize: .EQUATE 6 ; formal param
             ;rotVect: .EQUATE 8 ; formal param
01D7  580004 rotRight:SUBSP   4,i         ; push #vector #size
01DA  CB0006          LDWX    rotSize,s
01DD  780002          SUBX    2,i
01E0  C70008          LDWA    rotVect,sfx
01E3  E30002          STWA    tempVal,s

01E6  A80000 rotRLoop:CPWX    0,i
01E9  1401FE          BRLE    rotREnd

01EC  780002          SUBX    2,i
01EF  C70008          LDWA    rotVect,sfx
01F2  680002          ADDX    2,i
01F5  E70008          STWA    rotVect,sfx
01F8  780002          SUBX    2,i
01FB  1201E6          BR      rotRLoop

01FE  C30002 rotREnd: LDWA    tempVal,s
0201  E40008          STWA    rotVect,sf
0204  500004          ADDSP   4,i         ; pop #vector #size
0207  01              RET                 ;rotRight


             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; int malloc ()
             ;;
             ;; Takes int size via accumulator, and adds its
             ;; value to the heap pointer, thus reserving
             ;; room for the new object
             ;;
             ;; Returns the new object's address within the
             ;; heap, via the index register
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0208  C90340 malloc:  LDWX    hpPtr,d
020B  610340          ADDA    hpPtr,d
020E  E10340          STWA    hpPtr,d
0211  01              RET


             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
             ;; OUTPUT MESSAGES
             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

0212  486F77 sizeMsg: .ASCII  "How big is your vector? \x00"
      206269
      672069
      732079
      6F7572
      207665
      63746F
      723F20
      00
022B  0A0A45 cmdPrmpt:.ASCII  "\n\nEnter command.  L = left R = right E =
exchange Q = quit \x00"
```

```
      6E7465
      722063
      6F6D6D
      616E64
      2E2020
      4C203D
      206C65
      667420
      52203D
      207269
      676874
      204520
      3D2065
      786368
      616E67
      652051
      203D20
      717569
      742000
0267  0A4361 errExMsg:.ASCII  "\nCaution: Out of Bounds Exchange Attempted\n
\x00"
      757469
      6F6E3A
      204F75
      74206F
      662042
      6F756E
      647320
      457863
      68616E
      676520
      417474
      656D70
      746564
      0A00
0293  0A496E errInput:.ASCII  "\nIncorrect choice. Try again.\n\x00"
      636F72
      726563
      742063
      686F69
      63652E
      205472
      792061
      676169
      6E2E0A
      00
02B2  0A4578 xchngMsg:.ASCII  "\nExchange which 2 locations?\n\x00"
      636861
      6E6765
      207768
      696368
      203220
      6C6F63
      617469
      6F6E73
      3F0A00
02D0  0A4279 exitMsg: .ASCII  "\nBye bye\x00"
      652062
      796500
02D9  0A5B00 inVectP1:.ASCII  "\n[\x00"
02DC  5D3A20 inVectP2:.ASCII  "]: \x00"
      00
02E0  2000   spcIsSpc:.ASCII  " \x00"
02E2  0A00   newLine: .ASCII  "\n\x00"
```

```
              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
              ;; JUMP TABLE
              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

02E4  006C    choiceJT:.ADDRSS caseE       ; 'E' input
02E6  0063             .ADDRSS caseErr
02E8  0063             .ADDRSS caseErr
02EA  0063             .ADDRSS caseErr     ; 'H' input
02EC  0063             .ADDRSS caseErr
02EE  0063             .ADDRSS caseErr
02F0  0063             .ADDRSS caseErr     ; 'K' input
02F2  00A8             .ADDRSS caseL       ; 'L' input
02F4  0063             .ADDRSS caseErr
02F6  0063             .ADDRSS caseErr
02F8  0063             .ADDRSS caseErr     ; 'O' input
02FA  0063             .ADDRSS caseErr
02FC  00C3             .ADDRSS caseQ       ; 'Q' input
02FE  00DF             .ADDRSS caseR       ; 'R' input
0300  0063             .ADDRSS caseErr
0302  0063             .ADDRSS caseErr
0304  0063             .ADDRSS caseErr     ; 'U' input
0306  0063             .ADDRSS caseErr
0308  0063             .ADDRSS caseErr
030A  0063             .ADDRSS caseErr     ; 'X' input
030C  0063             .ADDRSS caseErr
030E  0063             .ADDRSS caseErr
0310  0063             .ADDRSS caseErr     ; '[' input
0312  0063             .ADDRSS caseErr
0314  0063             .ADDRSS caseErr
0316  0063             .ADDRSS caseErr     ; '^' input
0318  0063             .ADDRSS caseErr
031A  0063             .ADDRSS caseErr
031C  0063             .ADDRSS caseErr     ; 'a' input
031E  0063             .ADDRSS caseErr
0320  0063             .ADDRSS caseErr
0322  0063             .ADDRSS caseErr     ; 'd' input
0324  006C             .ADDRSS caseE       ; 'e' input
0326  0063             .ADDRSS caseErr
0328  0063             .ADDRSS caseErr
032A  0063             .ADDRSS caseErr     ; 'h' input
032C  0063             .ADDRSS caseErr
032E  0063             .ADDRSS caseErr
0330  0063             .ADDRSS caseErr     ; 'k' input
0332  00A8             .ADDRSS caseL       ; 'l' input
0334  0063             .ADDRSS caseErr
0336  0063             .ADDRSS caseErr
0338  0063             .ADDRSS caseErr     ; 'o' input
033A  0063             .ADDRSS caseErr
033C  00C3             .ADDRSS caseQ       ; 'q' input
033E  00DF             .ADDRSS caseR       ; 'r' input

              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
              ;; HEAP & HEAP POINTER
              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

0340  0342    hpPtr:   .ADDRSS heap
0342  00      heap:    .BLOCK  1
0343                   .END
-------------------------------------------------------------------------------
--
```

Symbol table

```
--------------------------------------
Symbol    Value        Symbol    Value
--------------------------------------
CaseEEnd  00A5         allocVec  0019
badChck   01A5         caseE     006C
caseErr   0063         caseL     00A8
caseQ     00C3         caseR     00DF
chInSize  FFFE         chckInpt  0183
chckSize  0002         choiceJT  02E4
cmdPrmpt  022B         errExMsg  0267
errInput  0293         exLoc1    0004
exLoc2    0006         exSize    0008
exVal     0000         exVect    000A
exchange  0140         exitMsg   02D0
false     0000         heap      0342
hpPtr     0340         inSize    0004
inVLoop   0100         inVect    00FA
inVectP1  02D9         inVectP2  02DC
inVector  0006         index     0000
inptLoc1  FFF8         inptLoc2  FFFA
inptSize  FFFC         inptVect  FFFE
mLoop     0038         main      0004
malloc    0208         newLine   02E2
prinLoop  012D         prinVect  0124
rotLEnd   01CA         rotLLoop  01B5
rotLeft   01A9         rotREnd   01FE
rotRLoop  01E6         rotRight  01D7
rotSize   0006         rotVect   0008
size      0000         sizeMsg   0212
spcIsSpc  02E0         tempVal   0002
true      0001         vector    0002
xchngEnd  017F         xchngMsg  02B2
--------------------------------------
```