

Problem 3. (32 pts) Shown below is the beginning of an implementation of the subroutine `Recursive` in Machine Problem #3. The inputs are indices `SI,DI` into `Array`. The outputs are $(SI,DI)=(r,s)$ with the maximum subsequence sum; `AX` is the value of this sum from r to s .

Code Segment	Offset in Stack Segment
<code>Recursive:</code>	
<code>8BFD CMP SI,DI</code>	0390 _____
<code>8BFF JNE .notbasecase</code>	
<code>8C01 SAL SI,1 ; Convert index to</code>	0392 _____
<code>8C03 MOV AX,[Array+SI] ; displacement</code>	
<code>8C07 SHR SI,1</code>	0394 _____
<code>8C09 RET</code>	
<code> .notbasecase: ; Here (SI) < (DI)</code>	0396 _____
<code>8C0A PUSH BP</code>	
<code>8C0B SUB SP,10 ; Allocate 5 words</code>	0398 _____
<code>8C0F MOV BP,SP ; for local variables</code>	
<code>8C11 MOV [BP+0],DI</code>	039A _____
<code>8C14 ADD DI,SI ; Middle index</code>	
<code>8C16 SHR DI,1 ; t=((SI)+(DI))/2</code>	039C _____
<code>8C18 MOV [BP+2],DI</code>	
<code>8C1B CALL Recursive ; Left subarray</code>	039E _____
<code>8C1E MOV [BP+4],SI ; Store best solution</code>	
<code>8C21 MOV [BP+6],DI ; seen so far</code>	03A0 _____
<code>8C24 MOV [BP+8],AX</code>	
<code>8C27 MOV SI,[BP+2] ; Set SI to t+1</code>	03A2 _____
<code>8C2A INC SI</code>	
<code>8C2B MOV DI,[BP+0] ; Original DI</code>	03A4 _____
<code>8C2E CALL Recursive ; Right subarray</code>	
<code>8C31 CMP AX,[BP+8] ; If right subarray</code>	03A6 _____
<code>8C34 JLE .notchange ; has larger sum</code>	
<code>8C36 MOV [BP+4],SI ; then replace best</code>	03A8 _____
<code>8C39 MOV [BP+6],DI ; solution seen so far</code>	
<code>8C3C MOV [BP+8],AX</code>	03AA _____
<code> .notchange:</code>	

(a) Suppose that just **before** a call to `Recursive`, `SP=03AE`, `SI=0004`, `DI=0007`, and values of the words at indices 4, 5, 6, 7 in `Array` are 0018, 0019, 001A, 001B respectively. (All values are in hexadecimal.) Determine the hexadecimal contents of words at addresses `SS:0390` through `SS:03AA` just **before** the execution of the `CMP SI,DI` instruction at `CS:8BFD` with `SI=DI=0005`. If there is insufficient information to determine a word, write “Unknown.”

(b) Assume that the `FixedEndSum` subroutine pushes only 3 words onto the stack. Suppose there are 50 (decimal) unused **words** on the stack just before an execution of `CALL Recursive`. Determine the maximum number of **words** (from indices `SI` to `DI`, inclusive) for which this call will not overflow the stack.

Problem 4. (30 pts) The subroutine `Enq` should implement a circular queue, as in Machine Problem #4. The code below has six logical mistakes. Explain each mistake and specify a correction, using the line numbers; a correction might change more than one statement.

```

KQList  RESB 0      ; Parameter list for Keyboard Queue
KQBeg   DW   KQArea ; Beginning offset of queue area
KQCap   DW   9      ; Capacity of queue
KQFront DW   0      ; Index of front item
KQRear  DW   0      ; Index of place to put next rear item
KQData  RESB 1      ; Byte to input or output
KQArea  RESB 9      ; Actually stores at most 8 data bytes
|
1      MOV  BX,[KQList] ; Enqueue byte in KQData onto keyboard queue
2      CALL Enq
|
3  Enq:
4      PUSHA
5      MOV  AX,[BX+6]
6      DEC  AX          ; (AX) = rear index - 1
7      CMP  AX,[BX+4]   ; Compare with front index and
8      JE   .exit       ; exit if queue is full
9      MOV  DI,[BX+6]   ; (DI) = rear index
10     MOV  AL,[BX+8]   ; Enqueue data byte
11     MOV  [DI],AL
12     MOV  AX,[BX+2]   ; (AX) = queue capacity
13     CMP  [BX+6],AX   ; If rear index is above
14     JBE  .exit       ; the capacity, then
15     MOV  WORD[BX+6],0 ; reset rear index to 0
16  .exit:
17     INC  WORD[BX+6]   ; Advance rear index
18     POPA
19     RET

```

1. _____

2. _____

3. _____

4. _____

5. _____

6. _____

Problem 5. (25 pts) Write your answers in clear English.

(a) In Machine Problem #3, suppose the starting time (hours : minutes : seconds . hundredths) is 6:18:57.00 and the ending time is 6:19:04.05. The `Elapsed` subroutine calls `binasc` and `dspmsg` twice and prints out 203.5 seconds instead of 7.05 seconds. Identify two mistakes in the code. (Hint: the 8-bit binary representation of 203 equals the 8-bit two's complement representation of -53.)

1. _____

2. _____

(b) In Machine Problem #4, suppose the ball has disappeared from the screen. The user presses the up-arrow key (\uparrow), and the ball reappears at its original location, but it does not move. All other functions work properly. Identify two possible causes for the incorrect behavior.

1. _____

2. _____

(c) In the `KbdInt` interrupt service routine in MP4, `DS` is set to equal to `CS`. `MOV AX, CS`
 What undesirable behavior might occur if these statements are omitted? Why? `MOV DS, AX`
