

Name(s): _____

Over the last couple of weeks, you have developed skills that are more valuable than you may think. Let's refine them. To read a character from the keyboard, you simply did the following...

```
MOV  AH,0h
INT  16H
```

...and the keystroke is stored into AL. This routine, with AH=0, waits for a key to be pressed within INT 16h prior to moving on. Now, recall that when AH=1, the INT 16h routine simply looks to see if a key has been pressed (clearing the zero flag if true) without waiting. Also recall that if a "special" key was struck (ALT-*something*, in our case), a 00h (zero) is returned in AL and the ASCII code for the special key (see handout) is stored in AH (e.g., ALT-F1 has an ASCII code of 68h in stored in AH).

Your task: Write a procedure that will query the keyboard once. Define the variable KEY to hold the ASCII code for a conventional keystroke and SPKEY to hold the ASCII code for a "special" keystroke, like ALT-F1. If no key has been pressed, it will return a 'FFh' (an invalid ASCII code) in both variables KEY and SPKEY. If a conventional key has been pressed, you must determine if it was special or not. If not, return a 'FFh' in SPKEY and the conventional ASCII code for the keystroke in KEY. If it was a special key, return the ASCII code for it in SPKEY and a 'FFh' in KEY. Note that this procedure allows key capture but need not halt a program from doing other things between keyboard queries – an important capability for complex algorithms.

Now, write a main program that tests your code – it should echo the keystroke to the screen and **continue** writing consecutive keystrokes to the screen. Halt the program when the user hits ALT-F1. **Note that since the keyboard returns the ASCII code for the key pressed, there's no conversion necessary for displaying it to the screen.**

DEMONSTRATE THIS PROGRAM _____ *sign-off*

Now, allow the user, while they are entering keystrokes, to BACKSPACE (08h) and re-enter their keystrokes, OVERWRITING the previous keystrokes displayed on the screen.

DEMONSTRATE THIS PROGRAM _____ *sign-off*

HINT: You need not store the keystrokes in an array – write to screen on the fly.

GRADUATE STUDENTS (OR UNDERGRADS EXTRA CREDIT): Now, modify your code to allow the user to BACKSPACE, but the newly typed characters must be INSERTED (they can not overwrite previously existing characters).

This function is normally toggled to via the “Insert” key on your keyboard.

DEMONSTRATE THIS PROGRAM _____ *sign-off*

SOLUTION: The following procedure will query to see if a key has been pressed. If it has, it will store that keystroke into a variable named KEY, unless it was a special key, in which case the keystroke is stored in SPKEY.

```

PROC          NEAR          GETKEY
              PUSHALL

;Check to see if a key has been pressed
              SUB           AH,AH
              INT          16h

;If no key was pressed, place 'Z' into KEY and SPKEY and return
              JZ           NOKEY

;Otherwise, grab the keystroke -
              MOV          AH,1h
              INT          16h

;Check to see if it was a "special key" (in which case AL=0)
;...if so, place AH into AL
              CMP          AL,0h
              JZ          SPECIAL

;Otherwise, place AL into KEY and 'Z' into SPKEY and return
              MOV          AH,'Z'
              JMP          FINISH

SPECIAL:     MOV          AL,'Z'
              JMP          FINISH

NOKEY:       MOV          AL,'Z'
              MOV          AH,'Z'

FINISH:      MOV          KEY,AL
              MOV          SPKEY,AH
              RET

ENDP        GETKEY

```

This routine will query the keyboard buffer once. If no key was pressed, it will return a 'Z' (5Ah) in variable KEY and SPKEY, otherwise, the pressed key will be returned (of course, this assumes 'Z' is a safe "no key hit" indicator for use in the main program). The main program would look something like...

```

AGAIN:       CALL          GETKEY
              MOV          AL,KEY
              CMP          AL,'Z'
              JZ          CHECKSP
              JMP          SERVICEKEY

CHECKSP:     CMP          AH,'Z'
              JZ          TRYAGAIN
              JMP          SERVICESP

TRYAGAIN:    JMP          AGAIN

```