ORG 100H

.MODEL SMALL

.STACK 100H

.DATA

buffer DB 100 dup('$')

todo\_text1 DB 100 dup('$')

todo\_text2 DB 100 dup('$')

todo\_date1 DB 100 dup('$')

todo\_date2 DB 100 dup('$')

taskAdded DB 0

msg\_header DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*",0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\* Welcome to, Task Assistant \*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "", 0DH, 0AH

DB " [1] Show all tasks", 0DH, 0AH

DB " [2] Add new task", 0DH, 0AH

DB " [3] Update a task", 0DH, 0AH

DB " [4] Delete a task", 0DH, 0AH

DB " [5] Search a task", 0DH, 0AH

DB " [0] Exit", 0DH, 0AH

DB "", 0DH, 0AH

DB "Press any of the given digit to continue...$"

msg\_show\_all DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\* All of your TASKS \*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "", 0DH, 0AH

DB "$"

msg\_update\_header DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\* UPDATE task \*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "", 0DH, 0AH

DB "$"

msg\_add\_header DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\* ADD NEW task \*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "", 0DH, 0AH

DB "$"

msg\_delete\_header DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\* DELETE A task \*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "", 0DH, 0AH

DB "$"

msg\_updating\_header DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\* Updating a task \*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "", 0DH, 0AH

DB "$"

msg\_search\_header DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\* SEARCH A TASK \*", 0DH, 0AH

DB "\* \*", 0DH, 0AH

DB "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", 0DH, 0AH

DB "", 0DH, 0AH

DB "$"

msg\_text DB "Enter Task description : $"

msg\_date DB "Enter Task date(DD/MM/YYYY): $"

msg\_added DB "Task added successfully...$"

msg\_delete DB " DELETE functionality goes here $"

msg\_empty DB "No task found$"

msg\_update DB "Enter the SL No of the task you want to UPDATE$"

msg\_option DB "[1] Main Menu [0] Exit$"

msg\_back DB "OR Press [0] to back$"

msg\_no\_match DB "No matching task found.$"

msg\_match\_found DB "Matching Task Found:$"

msg\_search\_prompt DB "Enter to search$", 0

newline DB 13, 10, "$"

SHOW MACRO message

MOV AH, 9

LEA DX, message

INT 21H

ENDM

SHOW\_HEADER MACRO

SHOW msg\_header

SHOW newline

ENDM

GETC MACRO

MOV AH, 1

INT 21H

ENDM

GET\_STRING MACRO var

MOV AH, 0Ah

LEA DX, buffer

INT 21H

; Copy the entered string (skip the first byte which is the length)

MOV SI, OFFSET buffer+2

LEA DI, var

MOV CX, 0

MOV CL, [buffer+1] ; Load the length of the entered string

REP MOVSB

; Terminate with `$`

MOV BYTE PTR [DI], '$'

ENDM

SHOW\_TASK MACRO sl\_no, var\_text, var\_date

PRINT ' '

PRINT sl\_no

PRINT '.'

SHOW var\_text

SHOW newline

PRINT ' '

PRINT ' '

PRINT ' '

PRINT ' '

SHOW var\_date

ENDM

SHOW\_ALL MACRO

ENDM

PRINT MACRO character

MOV AH, 2

MOV DL, character

INT 21H

ENDM

CLEAR\_SCREEN MACRO

MOV AH, 00

MOV AL, 02

INT 10H

ENDM

EXIT MACRO

MOV AX, 4CH

INT 21H

ENDM

SWAP\_STRINGS MACRO string1, string2, temp

; Clear the destination strings

XOR SI, SI

XOR CX, CX

MOV DI, OFFSET string1

REP STOSB

XOR SI, SI

XOR CX, CX

MOV DI, OFFSET string2

REP STOSB

; Copy contents from string1 to temp

MOV SI, OFFSET string1

MOV DI, OFFSET temp

MOV CX, 50

REPE MOVSB

; Copy contents from string2 to string1

MOV SI, OFFSET string2

MOV DI, OFFSET string1

MOV CX, 50

REPE MOVSB

; Copy contents from temp to string2

MOV SI, OFFSET temp

MOV DI, OFFSET string2

MOV CX, 50

REPE MOVSB

ENDM

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MAINMENU\_LOOP:

CLEAR\_SCREEN

SHOW\_HEADER

INPUT\_LOOP:

GETC

CMP AL, '1'

JE SM1

CMP AL, '2'

JE SM2

CMP AL, '3'

JE SM3

CMP AL, '4'

JE SM4

CMP AL, '5'

JE SM5

CMP AL, '0'

JE END\_MAIN

JMP INPUT\_LOOP

SM1:

CALL SHOW\_ALL\_TASKS

JMP MAINMENU\_LOOP

SM2:

CALL ADD\_NEW\_TASK

JMP MAINMENU\_LOOP

SM3:

CALL UPDATE\_TASK

JMP MAINMENU\_LOOP

SM4:

CALL DELETE\_TASK

JMP MAINMENU\_LOOP

SM5:

CALL SEARCH\_TASK

JMP MAINMENU\_LOOP

END\_MAIN:

EXIT

MAIN ENDP

SHOW\_ENTERED\_TASKS PROC

SHOW newline

CMP taskAdded, 0

JE EMPTY

SHOW\_TASK "1", todo\_text1, todo\_date1

CMP taskAdded, 2

JGE SECOND\_TASK\_OUTPUT

JMP CONT1

EMPTY:

SHOW msg\_empty

JMP CONT1

SECOND\_TASK\_OUTPUT:

SHOW newline

SHOW newline

SHOW\_TASK "2", todo\_text2, todo\_date2

JMP CONT1

; Point From EMPTY block

CONT1:

RET

SHOW\_ENTERED\_TASKS ENDP

SHOW\_ALL\_TASKS PROC

ALL\_TASKS\_LOOP:

CLEAR\_SCREEN

SHOW msg\_show\_all

;showing all the entered tasks

CALL SHOW\_ENTERED\_TASKS

SHOW newline

SHOW newline

SHOW msg\_option

SHOW newline

GETC

CMP AL, '1'

JE END\_SHOW\_ALL\_TASKS

CMP AL, '0'

JE ATL2

JMP ALL\_TASKS\_LOOP

ATL2:

EXIT

END\_SHOW\_ALL\_TASKS:

RET

SHOW\_ALL\_TASKS ENDP

ADD\_NEW\_TASK PROC

CLEAR\_SCREEN

SHOW msg\_add\_header

SHOW newline

SHOW msg\_text

MOV BL, taskAdded

CMP BL, 1

JE SECOND\_TASK

GET\_STRING todo\_text1

SHOW newline

SHOW msg\_date

GET\_STRING todo\_date1

JMP CONT

SECOND\_TASK:

GET\_STRING todo\_text2

SHOW newline

SHOW msg\_date

GET\_STRING todo\_date2

CALL SORT

CONT:

INC taskAdded

SHOW newline

SHOW newline

SHOW msg\_added

SHOW newline

SHOW newline

SHOW msg\_option

SHOW newline

GETC

RET

ADD\_NEW\_TASK ENDP

UPDATE\_TASK PROC

UPDATE\_TASK\_LOOP:

; clearing screen

CLEAR\_SCREEN

; showing header for update

SHOW msg\_update\_header

SHOW newline

SHOW msg\_update

SHOW newline

; Showing all tasks

CALL SHOW\_ENTERED\_TASKS

SHOW newline

SHOW newline

SHOW msg\_back

SHOW newline

GETC

CMP AL, '1'

JE UPDATE\_FIRST

CMP AL, '2'

JE UPDATE\_SECOND

CMP AL, '0'

JE END\_UPDATE\_TASK

JMP UPDATE\_TASK\_LOOP

UPDATE\_FIRST:

CLEAR\_SCREEN

SHOW msg\_updating\_header

SHOW newline

SHOW msg\_text

GET\_STRING todo\_text1

SHOW newline

SHOW msg\_date

GET\_STRING todo\_date1

CALL SORT

JMP UPDATE\_TASK\_LOOP

UPDATE\_SECOND:

CLEAR\_SCREEN

SHOW msg\_updating\_header

SHOW newline

SHOW msg\_text

GET\_STRING todo\_text2

SHOW newline

SHOW msg\_date

GET\_STRING todo\_date2

CALL SORT

JMP UPDATE\_TASK\_LOOP

END\_UPDATE\_TASK:

RET

UPDATE\_TASK ENDP

DELETE\_TASK PROC

DELETE\_TASK\_LOOP:

CLEAR\_SCREEN

SHOW msg\_delete\_header

SHOW newline

SHOW newline

; Showing all tasks

CALL SHOW\_ENTERED\_TASKS

SHOW newline

SHOW newline

SHOW msg\_back

SHOW newline

GETC

CMP AL, '0'

JE END\_DELETE\_TASK

CMP AL, '2'

JE DELETE\_SECOND

CMP AL, '1'

JE DELETE\_FIRST

JMP DELETE\_TASK\_LOOP

CMP taskAdded, 0

JE DELETE\_TASK\_LOOP

DEC taskAdded

JMP DELETE\_TASK\_LOOP

DELETE\_FIRST:

DEC taskAdded

JMP DELETE\_TASK\_LOOP

DELETE\_SECOND:

DEC taskAdded

JMP DELETE\_TASK\_LOOP

END\_DELETE\_TASK:

RET

DELETE\_TASK ENDP

SEARCH\_TASK PROC

CLEAR\_SCREEN

SHOW msg\_search\_header

SHOW newline

SHOW msg\_search\_prompt ; Display the prompt message

SHOW newline

GET\_STRING buffer

; Initialize a flag to track if a match is found

MOV AL, 0 ; 0 means no match found

; Compare with the first task if at least one task exists

CMP taskAdded, 0

JLE NO\_MATCH

LEA SI, todo\_text1

LEA DI, buffer

CALL COMPARE\_STRINGS

CMP AL, 1

JE MATCH\_FOUND1

; Compare with the second task if there are at least two tasks

CMP taskAdded, 1

JLE NO\_MATCH

LEA SI, todo\_text2

LEA DI, buffer

CALL COMPARE\_STRINGS

CMP AL, 1

JE MATCH\_FOUND2

; No match found

NO\_MATCH:

SHOW newline

SHOW msg\_no\_match

SHOW newline

SHOW msg\_back

GETC

RET

; Match found for the first task

MATCH\_FOUND1:

SHOW newline

SHOW msg\_match\_found

SHOW newline

SHOW\_TASK "1", todo\_text1, todo\_date1

SHOW newline

SHOW msg\_back

GETC

RET

; Match found for the second task

MATCH\_FOUND2:

SHOW newline

SHOW msg\_match\_found

SHOW newline

SHOW\_TASK "2", todo\_text2, todo\_date2

SHOW newline

SHOW msg\_back

GETC

RET

SEARCH\_TASK ENDP

COMPARE\_STRINGS PROC

MOV CX, 100 ; Limit the comparison length

MOV AL, 0 ; Default to no match

COMPARE\_LOOP:

LODSB ; Load a byte from the source string

SCASB ; Compare it with a byte from the destination string

JNE NOT\_EQUAL ; Exit loop if they don't match

CMP AL, '$' ; Stop comparing if end of string is reached

JE EQUAL

LOOP COMPARE\_LOOP

EQUAL:

MOV AL, 1 ; Match found

RET

NOT\_EQUAL:

MOV AL, 0 ; No match

RET

COMPARE\_STRINGS ENDP

SORT PROC

LEA SI, todo\_date1

LEA DI, todo\_date2

; Compare dates character by character

MOV CX, 10 ; Assuming dates have 10 characters

MOV AL, 0 ; Initialize result

L1:

MOV AH, [SI]

MOV BH, [DI]

CMP AH, BH

JE NextChar ; If characters are equal, go to the next character

; If characters are not equal, set result based on the comparison

JA Date1IsLater

Date1IsEarlier:

MOV AL, 1 ; is earlier

JMP EndComparison

Date1IsLater:

MOV AL, 2 ; is later

JMP EndComparison

NextChar:

INC SI

INC DI

LOOP L1

EndComparison:

CMP AL, 1

JE OCC\_EARLIER

CMP AL, 2

JE OCC\_LATER

RET

OCC\_EARLIER:

RET

OCC\_LATER:

CALL SWAP\_TODO

RET

SORT ENDP

SWAP\_TODO PROC

LEA SI, todo\_text1

LEA DI, todo\_text2

LEA BX, buffer

SWAP\_STRINGS todo\_text1, todo\_text2, buffer

LEA SI, todo\_date1

LEA DI, todo\_date2

SWAP\_STRINGS todo\_date1, todo\_date2, buffer

RET

SWAP\_TODO ENDP

END MAIN