23K-0032
Saif

Assignment 3

Date:

(Q1)

```
.data

    dividend  dword  0D4A4h
    divison   dword  0Ah
    threshold dword  5h

.code
main PROC
        mov eax, dividend
        mov ebx, divison

        call recursion

        exit
main ENDP

recursion PROC

        cmp eax, threshold
        jbe return

        push eax
        xor edx, edx
        div ebx
```

Date:

```
call recursion

pop eax

return:
    ret

recursion ENDP

END main
```

(Q2)

```
.data
    array dword 10,20,30,40,50,60,70,80 ; long array
    size dword 8

input byte "Enter the value, you want to
            search: ",0

msg1 byte "Value not found! ",0
msg2 byte "Value found at index: ",0
```

Date:

```
.code
main PROC
    mov edx, OFFSET input
    call writestring
    call readint
    mov ebx, eax

    push 0
    push ssize
    push ebx
    push OFFSET array

    INVOKE recursiveSearch, addr arr, ebx, 0, ssize
    cmp eax, -1
    je notfound

    mov edx, OFFSET msg2
    call writestring
    call writeint
    jmp return

notfound:
    mov edx, OFFSET msg1
    call writestring
```

Date:

```
return:
    [redacted] exit
    [redacted]
main ENDP

recursiveSearch PROC
    push ebp
    mov ebp, esp

    mov eax, [ebp+8]
    mov ebx, [ebp+12]
    mov ecx, [ebp+16]
    mov edx, [ebp+20]

    cmp ecx, edx
    jae notfound

    mov esi, eax
    mov eax, [esi + ecx* 4]
    cmp eax, ebx
    je found

    push edx
    push ebx
    mov eax, ecx
    inc eax
```

Date:

```
push eax
push esi

INVOKE recursiveSearch, esi, ebx, ecx+1, edx
jmp return

found:
    mov eax, ecx
    jmp return

not found:
    mov eax, -1

return:
    pop ebp
    ret

recursiveSearch ENDP

END main
```

(Q3)

```
.data
    initialstr  byte "This is the source string",0
    target      byte 100 DUP (0)
    length      dword 0

.code
main PROC
        mov esi, OFFSET initialstr
        mov edi, OFFSET target
        mov ecx, 0

next:
        mov al, [esi + ecx]
        cmp al, 0
        je  return

        push ecx
        call is_duplicate
        pop ecx

        cmp al, 1
        je skip

        mov edx, length
        mov bl, [esi + ecx]
```

Date:

```
mov [edi + edx], bl
inc length

skip:
    inc ecx
    jmp next

retun:
    mov edx, OFFSET target
    call writestring
    call crlf
    exit
main ENDP

is_duplicate PROC
    push esi
    push edi
    push ebx
    push ecx
    push edx

    mov al, [esi + ecx]
    mov edx, 0
```

Date:

```
L1:
    cmp   edx, length
    jae   notfound
    mov   bl, [target + edx]
    cmp   bl, al
    je    found

    inc   edx
    jmp   L1

found:
    mov   al, 1
    jmp   return

notfound:
    mov al, 0

return:
    pop   edx
    pop   ecx
    pop   ebx
    pop   edi
    pop   esi
    ret
is-duplicate ENDP
END   main
```

Date:

(Q4)

```
.data
    input byte 100 DUP(0)
    userinput byte "Enter a string: ",0
    vowelmsg byte "Vowels count: ",0

    countA dword 0
    countE dword 0
    countI dword 0
    countO dword 0
    countU dword 0

    msgA byte "A: ",0
    msgE byte "E: ",0
    msgI byte "I: ",0
    msgO byte "O: ",0
    msgU byte "U: ",0

.code
main PROC
    mov edx, offset userinput
    call writestring
    mov edx, offset input
    mov ecx, 100
    call readstring
    mov esi, offset input
```

Date:

```
L1:
    mov al, [esi]
    cmp al, 0
    je return

    cmp al, 'a'
    je incA
    cmp al, 'A'
    je incA

    cmp al, 'e'
    je incE
    cmp al, 'E'
    je incE

    cmp al, 'I'
    je incI
    cmp al, 'i'
    je incI

    cmp al, 'o'
    je incO
    cmp al, 'O'
    je incO

    cmp al, 'u'
    je incU
    cmp al, 'U'
    je incU

incA:
    inc countA
    jmp next

incE:
    inc countE
    jmp next

incI:
    inc countI
    jmp next

incO:
    inc countO
    jmp next

incU:
    inc countO
    jmp next
```

Date:

```
next:
    inc  esi
    jmp  L1

return:
        call  crlf
        mov   edx, OFFSET vowelmsg
        call  writestring

        mov edx, offset msgA
        call writestring
        mov eax, countA
        call writeint
        call crlf

        mov edx, offset msgE
        call writestring
        mov eax, countE
        call writeint
        call crlf

        mov  edx, offset msgI
        call writestring
        mov  eax, countI
        call writeint
        call crlf
```

Date:

```
mov edx, offset msg0
call writestring
mov eax, count0
call writeint
call crlf

mov edx, offset msg1
call writestring
mov eax, count1
call writeint
call crlf

exit

main ENDP
END main
```