1) Print a prompt asking for a number. Should except 0-99. Any value greater or less should cause an error message.

./hw6 runs the executable

The image below shows a prompt: Enter integer 0-99:

It takes user input

The –1, -2 cause an error message Invalid input 0-99 because they are less than zero.

The 105 causes an error message because it is over 99.

```
braylee@braylee-VirtualBox:~$ ./hw6
Enter integer 0-99: -1
Inavlid input 0-99
braylee@braylee-VirtualBox:~$ ./hw6
Enter integer 0-99: -2
Inavlid input 0-99
braylee@braylee-VirtualBox:~$ ./hw6
Enter integer 0-99: 105
Inavlid input 0-99
```

2) Convert input string to integer

This image is the part of the code that runs this. If you notice when I test bounds it tests them with an integer therefore, the string by that point has become an integer.

```
toInteger:
        cmp eax,2       ;if one digit input
        je ifTrue       ;if true jump to label ifTrue
        cmp eax,3       ;if 2 digit
        je ifTwo        ;if true jump to label ifTwo

ifTrue:
        mov byte [ecx+1],0x0    ;remove carriage return
        mov esi,[ecx+0]         ;put first digit in esi
        sub esi,0x30            ;sub 30 to turn to int
        ret

ifTwo:
        mov byte [ecx+2],0x0    ;remove carraige return
        mov dl,[ecx+0]          ;put first digit in dl
        mov eax,0x10
        sub dl,0x30             ;sub 0x30 to get int
        mul dl
        mov bl,[ecx+1]          ;put second digit in bl
        sub bl,0x30
        add al,bl               ;add two values together
        mov esi,eax             ;put val in esi
        ret

testBounds:
        cmp esi,-1
        jle errMsg
        cmp esi,0x99
        jg errMsg
        ret
```

3) Multiply the integer by ten and print the result to the terminal
    a. Multiply by ten

    The image below is the code I used to multiply by 10. It is commented in the code

```
toString:
        xor eax,eax
        mov edi,0               ;used as index counter
        mov ecx,0x30            ;will be used to add
        mov eax,0x10            ;multiply by 10
        mul esi
```

b. One digit

This is an example of one digit output. It uses the input 0 which comes out as 0 after it is multiplied by ten.

```
braylee@braylee-VirtualBox:~$ ./hw6
Enter integer 0-99: 0
0
braylee@braylee-VirtualBox:~$
```

c. Two digits

Here is an example of two-digit outputs. We have an input of 5 which becomes 50 when multiplied by 10. An input of 9 which has an output of 90 after being multiplied by 10. We have an input of 7 which outputs 70 after being multiplied by ten.

```
braylee@braylee-VirtualBox:~$ ./hw6
Enter integer 0-99: 5
50
braylee@braylee-VirtualBox:~$ ./hw6
Enter integer 0-99: 9
90
braylee@braylee-VirtualBox:~$ ./hw6
Enter integer 0-99: 7
70
```

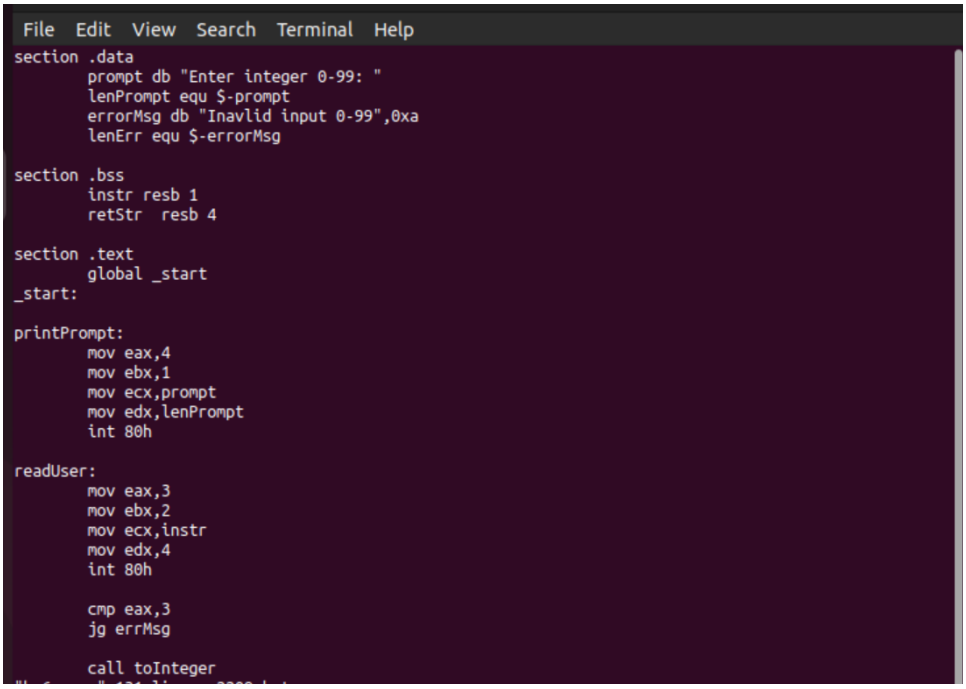
d. Three digits

These are the examples for a three-digit output. I entered 95 which becomes 950 as an output after being multiplied by ten. I input 90 which comes out as 900 after being multiplied by ten. I input 99 which becomes an output of 990 after being multiplied by 10.

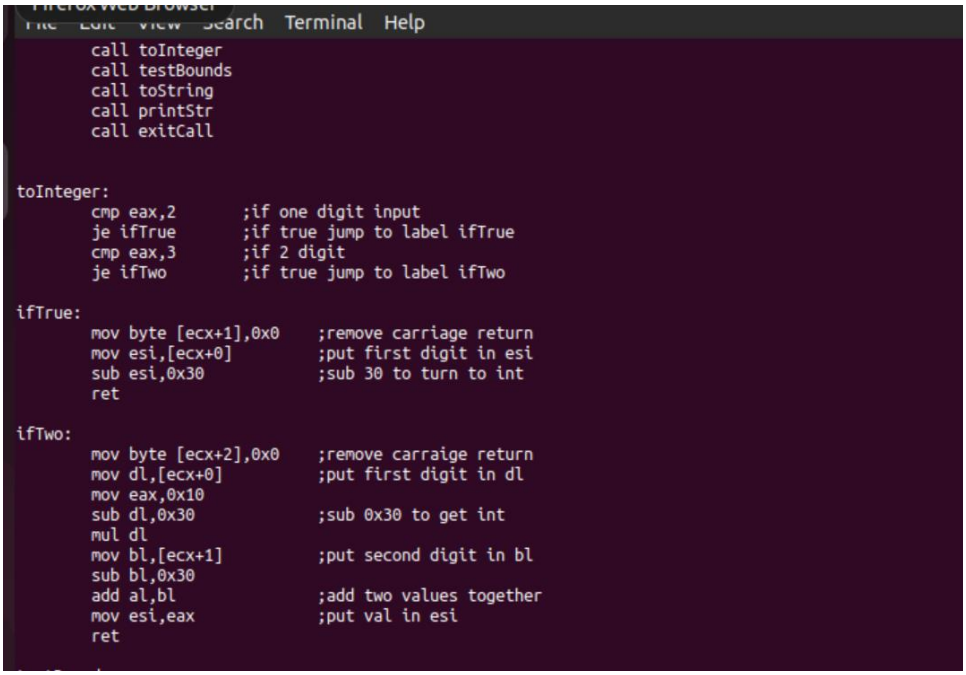
```
braylee@braylee-VirtualBox:~$ ./hw6
Enter integer 0-99: 95
950
braylee@braylee-VirtualBox:~$ ./hw6
Enter integer 0-99: 90
900
braylee@braylee-VirtualBox:~$ ./hw6
Enter integer 0-99: 99
990
```

4) The program exists. On all the examples above the program exists and you get the message braylee@braylee-VirtualBox:~$ which allows you to enter a new command to the terminal after the program runs.

**ScreenShot of asm:**



```
File  Edit  View  Search  Terminal  Help
section .data
        prompt db "Enter integer 0-99: "
        lenPrompt equ $-prompt
        errorMsg db "Inavlid input 0-99",0xa
        lenErr equ $-errorMsg

section .bss
        instr resb 1
        retStr  resb 4

section .text
        global _start
_start:

printPrompt:
        mov eax,4
        mov ebx,1
        mov ecx,prompt
        mov edx,lenPrompt
        int 80h

readUser:
        mov eax,3
        mov ebx,2
        mov ecx,instr
        mov edx,4
        int 80h

        cmp eax,3
        jg errMsg

        call toInteger
```



```
File  Edit  View  Search  Terminal  Help
        call toInteger
        call testBounds
        call toString
        call printStr
        call exitCall


toInteger:
        cmp eax,2       ;if one digit input
        je ifTrue       ;if true jump to label ifTrue
        cmp eax,3       ;if 2 digit
        je ifTwo        ;if true jump to label ifTwo

ifTrue:
        mov byte [ecx+1],0x0    ;remove carriage return
        mov esi,[ecx+0]         ;put first digit in esi
        sub esi,0x30            ;sub 30 to turn to int
        ret

ifTwo:
        mov byte [ecx+2],0x0    ;remove carraige return
        mov dl,[ecx+0]          ;put first digit in dl
        mov eax,0x10
        sub dl,0x30             ;sub 0x30 to get int
        mul dl
        mov bl,[ecx+1]          ;put second digit in bl
        sub bl,0x30
        add al,bl               ;add two values together
        mov esi,eax             ;put val in esi
        ret
```

```
File  Edit  View  Search  Terminal  Help
testBounds:
        cmp esi,-1
        jle errMsg
        cmp esi,0x99
        jg errMsg
        ret

errMsg:
        mov eax,4
        mov ebx,1
        mov ecx,errorMsg
        mov edx,lenErr
        int 80h
        call exitCall

toString:
        xor eax,eax
        mov edi,0               ;used as index counter
        mov ecx,0x30            ;will be used to add
        mov eax,0x10            ;multiply by 10
        mul esi
        mov ebx,0x100
        div ebx                 ;divide quotient in al remainder edx
        add eax,ecx
        mov byte[retStr+edi],al ;store quotient
        cmp al,0x30             ;check if first digit valid
        jne firstCheck
        secondDigit:
        mov eax,edx             ;move to remainder
        xor edx,edx
        mov ebx,0x10
        div ebx                 ;div 10 quotient in al remainder edx
```

```
File  Edit  View  Search  Terminal  Help
        mov ebx,0x10
        div ebx                 ;div 10 quotient in al remainder edx
        add eax,ecx
        mov byte[retStr+edi],al ;store quotient
        cmp al,0x30             ;check if second valid
        je secondCheck
        jne incIndex
        comeBack:
        add edx,ecx             ;change last remainder to string
        mov byte[retStr+edi],dl ;move into storage array
        inc edi
        mov byte[retStr+edi],0xa        ;add carraige return on end of line
        ret

firstCheck:
        inc edi         ;move index forward
        jmp secondDigit

secondCheck:
        cmp edi,0
        jne incIndex
        jmp comeBack

incIndex:
        inc edi         ;move index forward
        jmp comeBack
```

```
printStr:
        mov eax,4
        mov ebx,1
        mov ecx,retStr
        mov edx,32
        int 80h
        ret

exitCall:
        mov eax,1
        mov ebx,0
        int 80h
```

**Program Executing: (shows multiple outputs)**

The first image shows the nasm and ld lines as well.