Name: **Kashif Ali**
Roll No: **20P-0648**
Section: 3D
Lab-10 Tasks

```
INCLUDE Irvine32.inc
.data
array1 DWORD 10,6,7,3,2
count DWORD ?
.code
main PROC

    mov ECX, LENGTHOF array1; outer i
    DEC ECX

L3:
mov ESI, OFFSET array1  ;ESI now points to the first item of array1
     mov EDI, OFFSET array1+4
        mov count,ecx ; outer
        mov ECX, LENGTHOF array1 ; inner counter j
        DEC ECX

        L1:
                mov EAX, [ESI]
                mov EBX, [EDI]
                cmp EAX,EBX
                JBE LESS
                mov [ESI],EBX
                mov [EDI],EAX

        LESS:
             add ESI, TYPE array1
                 add EDI, TYPE array1
                 LOOP L1


         mov ecx,count ;outer counter
         LOOP L3

        mov ECX, LENGTHOF array1;here we just print the array
        mov ESI, OFFSET array1
L2:     MOV EAX, [ESI]
        call WriteInt
        call crlf
        add ESI, TYPE array1
        LOOP L2

exit

main ENDP
END main
```

Task 1 code

Output

```
+2
+3
+6
+7
+10
Press any key to continue . . .
```

try.asm

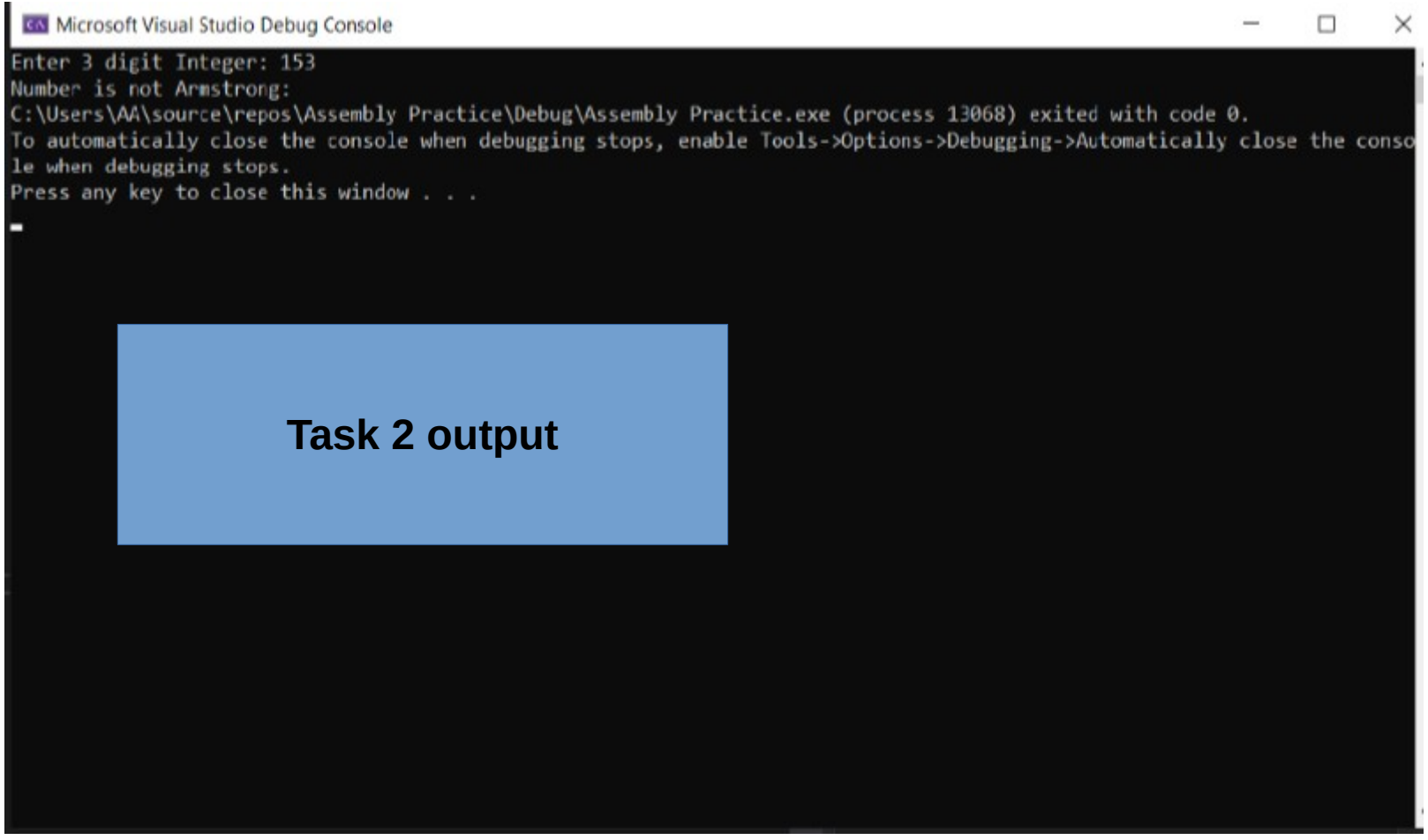
```
;Task 2

Include Irvine32.inc
.data
msg1 byte "Enter 3 digit Integer: ", 0
msg2 byte "Number is Armstrong: ", 0
msg3 byte "Number is not Armstrong: ", 0
.code
main PROC
call TakeInput
exit
main endp
TakeInput PROC Uses edx
    mov edx,offset msg1
    call writestring
    mov eax,0
    call readdec
    call Armstrong
    call display
ret
TakeInput ENDP
Armstrong PROC USES eax
    LOCAL remainder:byte , quotient:byte  , divisor:byte , numbers:byte  , cubes : dword
    mov esi,0
    mov ecx,3
    L1:
        mov divisor,10d
        div divisor
        mov quotient,al
        mov remainder[esi],ah
        inc esi
        mov eax,0
        movzx ax,quotient
    loop L1
    mov eax,0
    mov ecx,0
    mov ecx,3
    mov eax,0
    mov esi,0
    L2:
        mov al,remainder[esi]
        call calculate_cube
        mov cubes[esi * type cubes ],eax
        inc esi
        mov eax,0
    loop L2
```

Task 2 code part1

try.asm

```
47
48      mov ecx,0
49      mov ecx,3
50      mov esi,0
51      mov edx,0
52      L3:
53          add edx,cubes[esi * type cubes]
54          inc esi
55      loop L3
56
57  ret
58  Armstrong ENDP
59
60  calculate_cube PROC uses ecx
61      LOCAL  multiplier:byte
62
63      mov ecx,0
64      mov ecx,2
65      mov multiplier,al
66
67      L1:
68          mul multiplier
69      loop L1
70
71  ret
72  calculate_cube ENDP
73
74
75  display PROC
76  cmp eax,edx
77  JE display_equal
78
79  mov edx,offset msg2
80  call writestring
81  ret
82
83  display_equal:
84  mov edx,offset msg3
85  call writestring
86
87  ret
88  display ENDP
89  end main
```

Task 2 code part2


Microsoft Visual Studio Debug Console
Enter 3 digit Integer: 153
Number is not Armstrong:
C:\Users\AA\source\repos\Assembly Practice\Debug\Assembly Practice.exe (process 13068) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
Task 2 output

try.asm

```
1   INCLUDE Irvine32.inc
2   .data
3
4   msg BYTE "Enter The Number ", 0
5   msg2 BYTE "square of the Number is: ", 0
6
7   .code
8   main PROC
9   call squarenum
10  exit
11  main ENDP
12
13  squarenum PROC
14  ENTER 0,0
15  mov eax, 0
16  mov edx, 0
17  mov edx, offset msg
18  call WriteString
19  call crlf
20  call readDec
21  mov DWORD PTR[ebp-4],eax
22  mov eax, [ebp-4]
23  mul eax
24  mov edx, offset msg2
25  call WriteString
26  call crlf
27  call writeDec
28  ret
29
30  squarenum ENDP
31  END main
```

Task 4

output

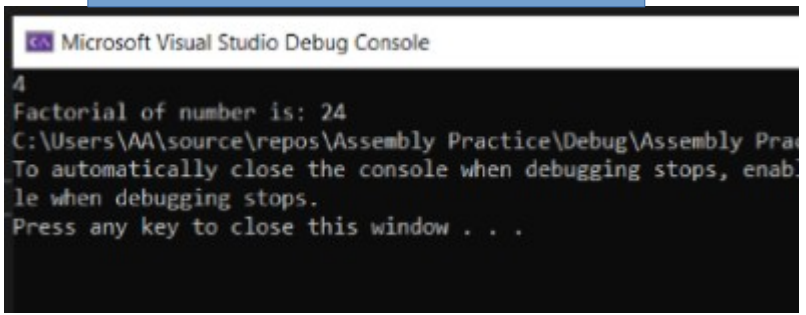
```
Enter The Number
4
Square Of The Number Is
16
Press any key to continue . . .
```

try.asm

```asm
;Task 5
Include Irvine32.inc
.code
msg1 byte "Factorial of number is: ", 0
main PROC

mov eax,0
call ReadDec
push eax
call Fact
mov edx, 0
mov edx, OFFSET msg1
call WriteString
call WriteDec
exit
main endp

Fact PROC
    enter 0,0
    mov ebx,0
    mov ebx,[ebp+8]
    cmp ebx,1
    JE ret_func
    JL ret_one
    dec ebx
    mul bl
    push ebx
    call Fact

    ret_func:
    leave
    ret 4

    ret_one:
    leave
    mov eax,1
    ret 4

Fact ENDP
end main
```

Task 5 code

Output



try.asm

```
;Task 6
Include Irvine32.inc
.data
not_prime_str byte 'Not A Prime Number',0
greatest_prime_str byte 'The largest prime number is : ',0
array DWORD 0,0,0
.code
main PROC

mov esi,0
mov ecx,4
L1:
    mov eax,0
    call readdec
    mov array[esi*type array],eax
    inc esi
    call checkPrime
loop L1


cmp eax,0
JE not_prime

call greatest_prime
exit

not_prime:
    mov edx,offset not_prime_str
    call writestring
    exit

main endp


greatest_prime PROC
    LOCAL max:dword
    mov max,0
    mov ecx,0
    mov ecx,lengthof array
    mov esi,0
    L3:
        mov eax,array[esi*type array]
        cmp eax,max
        JG update_max
        back_after:
        inc esi
    loop L3
```

Task 6 code part1

```
48
49
50
51      mov edx,offset greatest_prime_str
52      call writestring
53      mov eax,max
54      call writedec
55
56  ret
57
58  update_max:
59      mov max,eax
60      jmp back_after
61
62  greatest_prime ENDP
63
64
65  checkPrime PROC
66      LOCAL divisor:byte
67      mov divisor,2
68      div divisor
69      cmp ah,0
70      JE false_l
71      mov eax,1
72      ret
73
74      false_l:
75      mov eax,0
76      ret
77
78  checkPrime ENDP
79  end main
80
```

Task 6 code part 2

Task 6 output

Thank You

...