National University of Computer and Emerging Sciences, Lahore Campus

STUMM DAINERS	Course:	Computer Organization and Assembly Language	Course Code: Semester:	Fall 2021
THE COLUMN STATES	Program: Duration: Paper Date: Section(s): Exam:	BS (CS, DS) 60 Minutes 2-Dec-2021 All Midterm II	Total Marks: Weightage: Page(s): Section: Roll No:	30 15 8

Instruction/Notes:

- Exam is Open book, Open notes.
- Properly comment your code.
- You CANNOT use an instruction NOT taught in class.
- If there is any ambiguity, make a reasonable assumption. Questions during the exam are not allowed.
- Write your answer in the space provided. You can take extra sheets BUT they WON'T BE ATTACHED WITH THE QUESTION PAPER OR MARKED.
- All other rules pertaining to examinations as per NUCES policy apply.

Question 1 [15 Marks]: Short Questions

FAST School of Computing Page 1 of 8

Roll Number: _

Section:

[6 marks] Consider the following subroutine, which calculates the factorial of a number (size = 1 word). However, placed at the stack as a parameter and outputs the answer on the stack (size = 1 word). However, the placed at the stack as a parameter and outputs the answer of the functionality can be achieved, to code has some logical errors. Correct those errors so that the required functionality can be achieved. You can ADD or MODIFY existing lines, but you cannot REMOVE any line.

```
; Rewrite your code here
factorial:
   push bp
   mov bp, sp
   push ax
   push bx
    push dx
    mov ax, [bp+8]; copying the input
    cmp ax, 0
    ja L1
    mov word [bp+10], 1; returning the result
    jmp L2
 L1:
    sub sp, 2
    dec bp
     push bp; passing parameter for recursive subroutine
     call factorial; recursive subroutine call
 returnFact:
     pop bx
     mov dx, 0
     inc ax
     mul bx
     mov [bp+10], ax; returning the result
  L2:
     pop dx
     pop bx
      рор ах
      pop bp
      ret 6
```

₹	4-	
. '	word, the	
	1. 9	,
	(ye	
1	2	

Name: _ III.

Roll Number:	e out the sequence in which the instructions are umbered so your answer should be as follows:
Sample answer:	

Instructions executed in following order

111

110

You also have to briefly explain the working of this program.

	[org 0x0100]	Solution:
11	jmp start	
	The state of the s	
	my_rout:	
12	mov ax, 0x8434	
13	mov bl, 0x85	
4	div bl	
5	mov ax, 0xffff	1
6	mov dx, 0x0100	
7	mov bl, 0x3	
3	div bl	
9	ret	
	start:	
.0	call my_rout	
1	mov ax, 0x4c00	
2	int 0x21	

FAST School of Computing Page 3 of 8

movepixels: push ax push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) mov cx, 80 cld rep mov bx add si, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	rep movsb add si, 80 add di, 80 push bx push cx push si push di push es push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column)	movepixels: push ax push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov ds, ax mov di, 80 mov bx, 0 cld rep movsb add si, 80 add di, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	movepixels: push ax push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) cld rep movsb add si, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	novepixels: push ax push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov ds, ax mov di, 80 mov bx, 0 (code is continued in the second column) cld rep movsb add si, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	cld rep movsb add si, 80 add di, 80 push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) cld rep movsb add si, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push ax push bx push cx push si	rep movsb add si, 80
push ax push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) add si, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push ax push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov ds, ax mov di, 80 mov bx, 0 add si, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop bx pop ax ret	push ax push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) add si, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push ax push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) add si, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push ax push bx push cx push si push di push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) add si, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push ax push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) add si, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push ax push bx push cx push si	add si, 80
push bx push cx push si push di push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) add di, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop cx pop bx pop ax ret	push bx push cx push si push di push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) add di, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) add di, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push bx push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 mov bx, 0 ; (code is continued in the second column) add di, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push bx push cx push si push di push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) add di, 80 add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push bx push cx push si	
push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push cx push di push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov ds, ax mov di, 80 mov bx, 0 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret	push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels mov ax, 0x4c00	push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push si push di push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push cx push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push cx push si	add di, 80
push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop bx pop ax ret	push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push si push di push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push si push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) add bx, 1 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push si	
push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop si pop cx pop bx pop bx pop ax ret	push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push di push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) cmp bx, 25 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels		
push es push ds mov ax, 0xb800 mov es, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 jne loop1 pop ds pop es pop di pop si pop cx pop bx pop bx pop ax ret	push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	push es push ds mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels mov ax, 0x4c00	push es push ds mov ax, 0xb800 mov es, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels mov ax, 0x4c00	push es push ds mov ax, 0xb800 mov es, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) jne loop1 pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels mov ax, 0x4c00		add bx, 1
mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	mov ax, 0xb800 pop ds mov es, ax pop di mov ds, ax pop si mov si, 0 pop cx mov di, 80 pop bx mov bx, 0 pop ax ret	mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels mov ax, 0x4c00	push di	cmp bx, 25
mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	mov ax, 0xb800 pop es mov es, ax pop di mov ds, ax pop si mov si, 0 pop cx mov di, 80 pop bx mov bx, 0 pop ax ret	mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels	mov ax, 0xb800 mov es, ax mov ds, ax mov di, 80 mov bx, 0 ; (code is continued in the second column) pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels mov ax, 0x4c00	mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) pop ds pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels mov ax, 0x4c00	push es	jne loop1
mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 pop es pop di pop si pop cx pop bx pop bx pop ax ret	mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels mov ax, 0x4c00	mov ax, 0xb800 mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) pop es pop di pop si pop cx pop bx pop ax ret start: call movepixels mov ax, 0x4c00	push ds	
mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov es, ax pop di pop si pop si pop cx pop cx pop bx pop bx pop ax ret ; (code is continued in the second column)	mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov es, ax mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00		pop ds
mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov ds, ax mov si, 0 mov di, 80 mov bx, 0 code is continued in the second column)	mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov ds, ax mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00		
mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov si, 0 mov di, 80 mov bx, 0 pop cx pop bx pop ax ret	mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov si, 0 mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00		pop di
mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov di, 80 mov bx, 0 pop bx pop ax ret ; (code is continued in the second column)	mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov di, 80 mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00		pop si
mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov bx, 0 pop ax ret ; (code is continued in the second column)	mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov bx, 0 ; (code is continued in the second column) start: call movepixels mov ax, 0x4c00		рор сх
; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	; (code is continued in the second column)	; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	; (code is continued in the second column) start: call movepixels mov ax, 0x4c00		pop bx
start: call movepixels mov ax, 0x4c00	; (code is continued in the second column)	; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	start: call movepixels mov ax, 0x4c00	; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	; (code is continued in the second column) start: call movepixels mov ax, 0x4c00	mov bx, o	pop ax
start: call movepixels mov ax, 0x4c00		start: call movepixels mov ax, 0x4c00	start: call movepixels mov ax, 0x4c00	start: call movepixels mov ax, 0x4c00	start: call movepixels mov ax, 0x4c00	: (code is continued to at	ret
call movepixels mov ax, 0x4c00	start:	call movepixels mov ax, 0x4c00	call movepixels mov ax, 0x4c00	call movepixels mov ax, 0x4c00	call movepixels mov ax, 0x4c00	(code is continued in the second column)	
mov ax, 0x4c00		mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00		
	call movepixels						call movepixels
	may av 0 4 00						may av 0-1-00
		int 0v21	III OXZI	III OXZI	III 0X21		
		III 0X21					IIIL OXZI
	mov av 0×4×00	int 0x21			1		
	mov av 0v4c00						mov ax, 0x4c00
	1110V dx, UX4CUU	int 0x21					
int 0x21	mov av 0v4c00		int 0x21	int 0x21	int 0x21		mov ax, 0x4c00
int 0x21	mov av 0×4c00		int 0x21	int 0x21	int 0x21		
Int 0x21	mov av Ov4c00		int 0x21	Int 0x21	int 0x21		
	mov av 0v4c00						mov av .0v4c00
							· ·
mov ax, 0x4c00		mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00		
mov ax, 0x4c00		mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00		
mov ax, 0x4c00	call movepixels	mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00		call movepixels
mov ax, 0x4c00		mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00		
mov ax, 0x4c00	call movepixels	mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00	mov ax, 0x4c00		call movepixels
	call movepixels						call movepixels
	mov av 0v4c00						mov ax. 0x4c00
int 0x21	mov av 0v4c00		int 0x21	int 0x21	int 0x21		mov ax, 0x4c00
int 0x21	mov av 0×4c00		int 0x21	int 0x21	int 0x21		
		int 0v21	111t 0X21	III OXZI	1110 0 221		
		int 0x21					int 0x21
					I I		

FAST School of Computing Page 4 of 8

Name:	Roll Number:	Section:	
Question 2 [15 Marks]:	Draw a triangle with two given poin	ts i.e. A (x1, y1) and B (x2, y2).	

- i. [3 Marks] Triangle must be isosceles (two sides equal) and right (one 90-degree angle), for that purpose check two conditions given below:
 - a) y1 must be less than y2 and x1 must be less than x2.
 - b) (x2-x1) must be equal to (y2-y1).

No need to check other conditions as these two conditions are enough.

- ii. [2 Marks] Clear screen with white background.
- iii. [7 Marks] Only print the boundary of the triangle with red color and asterisk character (ASCII= 2A-Hex,42-Decimal).

Hint: Write a generic subroutine to print an asterisk on a single point. Use loops to print borders.

iv. [3 Marks] Write a program with proper subroutine names and stack implementation is compulsory for parameter passing.

<u>Note:</u> You can't use software interrupts. You should use hard code inputs but functions should be generic. It should run properly on any inputs.

Example 1:

Input: A (7, 8) and B (10, 11)

Output:

(7,8)

: •

(10,11)

Example 2:

Input: A (10, 11) and B (7, 8) Output: No printing on screen

Example 3:

Input: A (7, 8) and B (10, 10) Output: No printing on screen