**Lab Assignment # 01**

**COAL LAB – Spring 2021**

**Submission Deadline: Sunday 31st March 2022 @ 11:59 PM**

**Instructions Total Marks: 50**

| a. Programs must be MASM615 compatible.  b. Code must be commented properly.  c. For Part1, you have to write your answers in the space provided in this **.docx file.** d. For Part2, You have to submit all your codes in a **single .zip file;**  e. Submitted Filename must have format like **Assignment#\_Sec#\_Roll#** e.g. A1\_A\_19\_i1234 f. **Plagiarism will result in ZERO Marks in All Assignments of Class and Lab.** |
| --- |

**Q1.** Write an assembly program to implement the Number Guessing Game. In this game, first a user will close his eyes and then he will enter any number between 0-9. As the program will take that input through 07h so user won’t be sure about the number he entered [hopefully]. ( **10 Marks )**

Now he starts to guess the number within three attempts.

- If he guesses the number successfully print **‘W’** and exit the program - If he guesses the number wrong print **‘F’** in the first two attempts

- If he wrongly guesses the number for three consecutive times display **‘L’**

**Note:** Make sure you take all the inputs through **07h**, so that the user with short memory power can have more fun.

**Q2 .** Input a list of Strings **e.g ['Ali', 'Usman', 'Abdullah']** You have to sort them in ascending order . (Take Input a size of array). ( **10 Marks )**

To decide which string is greater use the following algorithm. Multiply the ascii of each character with its index and add them. The string with higher value would be greater.

For Example suppose the string passed are Ali & Umar then :

Ali : (ascii of A) \* 1 + (Ascii of l) \* 2 + (Ascii of i) \* 3 = (65) \* 1 + (108) \* 2 + (105) \* 3 = **491**

Umar : (ascii of U) \* 1 + (Ascii of m) \* 2 + (Ascii of a) \* 3 + (Ascii of r) \* 4 = **(85)** \* 1 + (109) \* 2 + (97) \* 3 + (114) \*4 = **1050**

**Q3.** Write down the equivalent code in assembly. **( 10 Marks )**

#include<iostream>

using namespace std;

int main()

{

int a , b , c , d, e , f , g ;

cin >> a >> b >>c>>d>>e>>f>>g; if

((g==f || e!=d) && (g==d || f<=e))

{ if(a>g && b <=e ){

if( c>a && e >= c){ cout<<a;

}

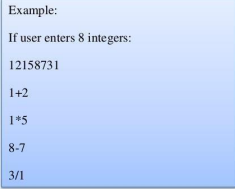
}

}

return 0;

}

**Q4.** Take 8 integers from the user and apply following operation on them. **ADD, SUB, MUL, DIV** Apply operation on two consecutive integers, Store result of each variable in separate variable. **( 10 Marks )**

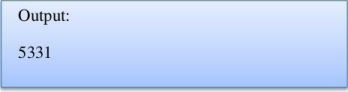
****

Sum = 3

Subtraction = 1

Multiply = 5 Divide

= 3

Now arrange the results in descending order and display them on the console. 

**Q5.** Print following pattern by taking input from user (make sure user enters odd number): **( 10 Marks )**

E.g. if user enters 5 print following pattern. 5 can be number of lines or maximum number of stars, it totally depends on your logic.

**\***

**\*\*\***

**\*\*\*\*\***

**\*\*\***

**\***

**Q1**

**.model small**

**.stack 100h**

**.data**

**guess db ?**

**input db ?**

**.code**

**mov ax ,@data**

**mov ds ,ax**

**;--------------------**

**; input to guess**

**mov ah, 07h**

**int 21h**

**sub al,48**

**mov guess, al**

**;--------------------**

**; second input**

**mov ah, 01h**

**int 21h**

**sub al,48**

**mov input, al**

**;--------------------**

**mov al, guess**

**cmp al, input**

**je W ; success attempt**

**jne FL ; 1st attempt wrong, 1 left**

**W:**

**;print W**

**mov dl,'W'**

**mov ah, 02h**

**int 21h**

**FL:**

**; 2nd attempt**

**mov ah, 01h**

**int 21h**

**sub al,48**

**mov input, al**

**je F ; success attempt**

**jne L ; 2nd attempt wrong, 0 left**

**F:**

**;print F**

**mov dl,'F'**

**mov ah, 02h**

**int 21h**

**L:**

**;print L**

**mov dl,'L'**

**mov ah, 02h**

**int 21h**

**Q2**

**.model small**

**.stack 100h**

**.data**

**var1 db ?,?,?,?**

**result db 0,0,0,0**

**.code**

**mov ax ,@data**

**mov ds ,ax**

**mov cx, SIZEOF var1**

**mov dx, OFFSET var1**

**mov ah, 3Fh**

**int 21h**

**input:**

**mov dx, OFFSET var1[si]**

**mov ah, 3Fh**

**int 21h**

**inc si**

**loop input**

**mov cx, SIZEOF var1**

**multiply:**

**mov al, var1**

**mov bl, 1**

**mul bl**

**add result, al**

**mov al, var1**

**mov bl, 2**

**mul bl**

**add result, al**

**mov al, var1**

**mov bl, 3**

**mul bl**

**add result, al**

**mov al, var1**

**mov bl, 4**

**mul bl**

**add result, al**

**loop multiply**

**mov al, var1**

**cmp al, var2**

**cmp al, var3**

**jg L1**

**jmp exit**

**mov al, var2**

**cmp al, var1**

**cmp al, var3**

**jg L2**

**jmp exit**

**mov al, var3**

**cmp al, var1**

**cmp al, var2**

**jg L3**

**jmp exit**

**L1:**

**L2:**

**L3:**

**end**

**Q3**

**.model small**

**.stack 100h**

**.data**

**; int a , b , c , d, e , f , g ;**

**a db ?**

**b db ?**

**c db ?**

**d db ?**

**e db ?**

**f db ?**

**g db ?**

**.code**

**start:**

**mov ax,@data**

**mov ds,ax**

**; int a , b , c , d, e , f , g ;**

**; cin >> a >> b >>c>>d>>e>>f>>g;**

**; if((g==f || e!=d) && (g==d || f<=e))**

**; {**

**; if(a>g && b <=e )**

**; {**

**; if( c>a && e >= c)**

**; {**

**; cout<<a;**

**; }**

**; }**

**; }**

**; return 0;**

**; cin >> a >> b >>c>>d>>e>>f>>g;**

**; take input a**

**mov ah, 01h**

**int 21h**

**sub al,48**

**mov a, al**

**; take input b**

**mov ah, 01h**

**int 21h**

**sub al,48**

**mov b, al**

**; take input c**

**mov ah, 01h**

**int 21h**

**sub al,48**

**mov c, al**

**; take input d**

**mov ah, 01h**

**int 21h**

**sub al,48**

**mov d, al**

**; take input e**

**mov ah, 01h**

**int 21h**

**sub al,48**

**mov e, al**

**; take input f**

**mov ah, 01h**

**int 21h**

**sub al,48**

**mov f, al**

**; take input g**

**mov ah, 01h**

**int 21h**

**sub al,48**

**mov g, al**

**; ((g==f || e!=d) && (g==d || f<=e))**

**mov al, g**

**cmp al, f**

**je L1**

**mov al, e**

**cmp al, d**

**jne L1**

**jmp exit**

**L1:**

**mov al, g**

**cmp al, d**

**je L2**

**jmp exit**

**mov al, f**

**cmp al, e**

**jg L2**

**ja L2**

**jmp exit**

**L2:**

**;if(a>g && b<=e)**

**; a>g**

**mov al, a**

**cmp al, g**

**jg L3**

**ja L3**

**jmp exit**

**L3:**

**;b<=e**

**mov al, b**

**cmp al, e**

**jge L4**

**jae L4**

**jmp exit**

**L4:**

**; (c>a && e >= c)**

**; c>a**

**mov al, c**

**cmp al, a**

**jg L5**

**ja L5**

**jmp exit**

**L5:**

**;e >= c**

**mov al, e**

**cmp al, c**

**jae L6**

**jge L6**

**jmp exit**

**L6:**

**; if( c>a && e >= c)**

**; {**

**; cout<<a;**

**; }**

**; c>a**

**mov al, c**

**cmp al, c**

**jae L7**

**jge L7**

**jmp exit**

**L7:**

**; cout << a;**

**mov al, a**

**mov dl, al**

**add dl, 48**

**mov ah, 02**

**int 21h**

**exit:**

**end start**

**Q4**

**.model small**

**.386**

**.stack 100h**

**.data**

**array DB 8 DUP(?)**

**var\_add DB ?**

**var\_mul DB ?**

**var\_sub DB ?**

**var\_div DB ?**

**.code**

**start:**

**mov ax, @data**

**mov ds, ax**

**mov si, OFFSET array**

**mov cx, SIZEOF array**

**input:**

**mov ah, 01h**

**int 21h**

**sub al,48**

**mov array[si], al**

**inc si**

**loop input**

**mov si, OFFSET array**

**mov cx, SIZEOF array**

**;-----------------------------------------------------------------------------**

**Operations:**

**; a =**

**; {**

**; 0001, 0002, si+0 --> 0 :4**

**; 0003, 0004, si+2 --> 5 :8**

**; 0005, 0006, si+4 --> 9 :12**

**; 0007, 0008, si+6 --> 13:16**

**; }**

**addition:**

**; si = 0**

**; array[si+0] + array[si+1] OR array[si+0] + array[si+4]**

**mov al, array[si]**

**mov bl, array[si+4]**

**;-----------------------**

**add al, bx**

**mov var\_add, al**

**;-----------------------**

**inc si**

**multiplication:**

**; si = 1**

**; array[si+2] \* array[si+2+1] OR array[si+5] \* array[si+5+4]**

**mov al, array[si]**

**mov bl, array[si+4]**

**;-----------------------**

**mul bl**

**mov var\_mul, al**

**;-----------------------**

**inc si**

**subtraction:**

**; si = 2**

**; array[si+4] - array[si+4+1] OR array[si+10] - array[si+10+4]**

**mov al, array[si]**

**mov bl, array[si+4]**

**;-----------------------**

**sub al, bl**

**mov var\_sub, al**

**;-----------------------**

**inc si**

**division**

**; si = 3**

**; a[si+6] / a[si+6+1] OR a[si+15] / a[si+6+4]**

**mov al, array[si]**

**mov bl, array[si+4]**

**;-----------------------**

**div bl**

**mov var\_div, al**

**;-----------------------**

**inc si**

**;-----------------------------------------------------------------------------**

**Descending\_Sort:**

**; var\_add**

**; var\_sub**

**; var\_mul**

**; var\_div**

**;var\_add**

**;-----------------------**

**mov al, var\_add**

**cmp al, var\_add**

**cmp al, var\_sub**

**cmp al, var\_mul**

**cmp al, var\_div**

**jb L1**

**jl L1**

**;-----------------------**

**;var\_sub**

**;-----------------------**

**mov al, var\_sub**

**cmp al, var\_add**

**cmp al, var\_sub**

**cmp al, var\_mul**

**cmp al, var\_div**

**jb L2**

**jl L2**

**;----------------------**

**;var\_mul**

**;-----------------------**

**mov al, var\_mul**

**cmp al, var\_add**

**cmp al, var\_sub**

**cmp al, var\_mul**

**cmp al, var\_div**

**jb L3**

**jl L3**

**;-----------------------**

**;var\_div**

**;-----------------------**

**mov al, var\_div**

**cmp al, var\_add**

**cmp al, var\_sub**

**cmp al, var\_mul**

**cmp al, var\_div**

**jb L4**

**jl L4**

**;-----------------------**

**;var\_add**

**L1:**

**add al,48**

**mov dl,al**

**mov ah,02h**

**int 21h**

**jmp exit:**

**;var\_sub**

**L2:**

**add al,48**

**mov dl,al**

**mov ah,02h**

**int 21h**

**jmp exit:**

**;var\_mul**

**L3:**

**add al,48**

**mov dl,al**

**mov ah,02h**

**int 21h**

**jmp exit:**

**;var\_div**

**L4:**

**add al,48**

**mov dl,al**

**mov ah,02h**

**int 21h**

**jmp exit:**

**;-----------------------------------------------------------------------------**

**exit:**

**mov ah,4ch**

**int 21h**

**end start**

**Q5**

**.model small**

**.stack 100h**

**.data**

**string db "Please enter the number =",'$'**

**space dw 11**

**star dw 1**

**space1 dw 1**

**star1 dw 13**

**input dw 0**

**downward dw 0**

**startToMultiply dw 2**

**maxSpace dw 11**

**.code**

**main proc**

**mov ax,@data**

**mov ds,ax**

**mov ah,09h**

**mov dx,offset string**

**int 21h**

**mov ax,0**

**mov ah,01h**

**int 21h**

**mov ah,0**

**mov bx,0**

**mov bx,ax**

**sub bl,48**

**mov ah,02h**

**mov dl,0dh**

**int 21h**

**mov dl,0ah**

**int 21h**

**mov ah,0**

**mov cx,0**

**mov input,bx**

**mov cx,bx**

**outer:**

**mov bx,0**

**mov bx,cx ;store**

**mov cx,space**

**k:**

**mov dl,32 ;space print**

**mov ah,02**

**int 21h**

**loop k**

**dec space; next time decrement**

**mov cx,star**

**l:**

**mov dl,'\*' ;space stars**

**mov ah,02**

**int 21h**

**loop l**

**inc star**

**inc star**

**mov ah,02h**

**mov dl,0dh**

**int 21h**

**mov dl,0ah**

**int 21h**

**mov cx,bx**

**loop outer**

**sub input,1**

**mov ax,0**

**mov ax,input**

**mov downward,ax**

**mov ax,0**

**mov ax,downward**

**mul startToMultiply**

**mov downward,ax**

**sub downward,1**

**;idher downward = 2\*input-1**

**;mov ax,0**

**;;mov ax,input**

**;mul startToMultiply**

**mov ax,0**

**mov ax,downward**

**mov star1,ax**

**mov ax,0**

**add space1,5**

**;space = maxSpace - input + 1**

**mov ax,0**

**mov ax,input**

**add ax,1**

**sub maxSpace,ax**

**add maxSpace,1**

**mov ax,0**

**mov ax,maxSpace**

**mov space1,ax**

**add space1,1**

**mov ax,0**

**mov cx,input**

**outer1:**

**mov bx,cx ;store**

**mov cx,space1**

**k1:**

**mov dl,32 ;space print**

**mov ah,02**

**int 21h**

**loop k1**

**inc space1; next time decrement**

**mov cx,star1**

**l1:**

**mov dl,'\*' ;space stars**

**mov ah,02**

**int 21h**

**loop l1**

**dec star1**

**dec star1**

**mov ah,02h**

**mov dl,0dh**

**int 21h**

**mov dl,0ah**

**int 21h**

**mov cx,bx**

**loop outer1**

**main endp**