```
include irvine32.inc
.data
stt1 byte "enter your name: ",0
myvar dd 20 dup(?)
stt2 byte "enter the col: ",0
stt3 byte "enter the row: ",0
  array dword 10 dup(?)
      rowlen = $ - array
      dword 10 dup(?)
      dword 10 dup(?)
      dword 10 dup(?)
      dword 10 dup(?)
      dword 10 dup(?)
      dword 10 dup(?)
      dword 10 dup(?)
      dword 10 dup(?)
      dword 10 dup(?)
       indexsize = type array
str1 byte "  ",0
str2 byte " |",0
space byte "************************",0
disp1 byte "enter row : ",0
disp2 byte "enter col : ",0
disp3 byte "you can only swap adjecent rows and colms",0
disp4 byte "row and column cannot be greater or smaller then the given matrix ",0
row1 dd ?
col1 dd ?
row2 dd ?
col2 dd ?
var5 dd ?
valindex1 dd ?
valindex2 dd ?
valindex3 dd ?
valindex4 dd ?
valclm  dd ?
temp dd ?
temp1 dd ?
temp2 dd ?
temp3 dd ?
tempval dd ?
tempval1 dd ?
tempval2 dd ?
vempval3 dd ?
tempva dd ?
tempva1 dd ?
tempva2 dd ?
vempva3 dd ?
tempindex1 dd ?
tempindex2 dd ?
tempindex3 dd ?
tempindex4 dd ?
tempindex5 dd ?
tempindex6 dd ?
tempinde1 dd ?
tempinde2 dd ?
```

```
tempinde3 dd ?
tempinde4 dd ?
tempinde5 dd ?
tempinde6 dd ?
scoredisp1 byte "                        SCORE : ",0
tempi dd ?
tempi1 dd ?
tempi2 dd ?
tempi3 dd ?
score dword 0
reminder dd 0
level2 dword 9,9,9,2,2,2,2,9,9,9
     rowlen2 = $ - level2
     dword 9,9,9,2,2,2,2,9,9,9
     dword 9,9,9,2,2,2,2,9,9,9
     dword 9,9,9,2,2,2,2,9,9,9
     dword 2,2,2,2,2,2,2,2,2,2
     dword 2,2,2,9,9,9,2,2,2,2
     dword 2,2,2,9,9,9,2,2,2,2
     dword 9,9,9,2,2,2,2,9,9,9
     dword 9,9,9,2,2,2,2,9,9,9
     dword 9,9,9,2,2,2,2,9,9,9
      indexsize2 = type array
        dispnotswap byte "this value cannot be swapped",0
        dp byte  "                          WELCOME TO NUMBER CRUSHER",0
        dp1 byte "                          TO PLAY (ENTER 1)",0
        dp2 byte "                          TO QUIT (ENTER 2)",0
        dp3 byte "                          Are you sure",0
        dp4 byte "                          yes (press 2)",0
        dp5 byte "                          NO (press 3) ",0
        dp6 byte "                       We hope you will comeback soon",0
        dispnam byte "NAME: ",0
        namee byte 20 dup(?)
        nameecount dword ?
        moves dword 0
        spa byte "            MOVES: ",0
        strr2 byte "bytes written to file [output.txt]: ",0
        showScoreStr db "00000",0
filename byte "output.txt",0
filehandle dword ?
ldisp1 byte "Abdullah",0
ldisp2 byte "300",0
lDISP3 BYTE 20 DUP(?)
lDISP4 BYTE 25 DUP(?)
lcountdisp1 dword ?
lcountdisp2 dword ?
NOOFBYTES DWORD ?
scorefromfile byte   3 dup(?)
highscore byte "THE CURRENT HIGH SCORE IS : ",0
varwithval dword 300
valGreater byte "Your score is greater", 0ah, 0dh, 0
valSmaller byte "Your score is less", 0ah, 0dh,0

.code
;this function will display a specifice string
```

```
disp33 proc
mov eax,brown(black*16)
call settextcolor
mov edx,offset disp3
call writestring
call crlf
mov eax,yellow(black*16)
call settextcolor
ret
disp33 endp
;this function will display a specifice string
disp44 proc
mov eax,brown(black*16)
call settextcolor
mov edx,offset disp4
call writestring
call crlf
mov eax,yellow(black*16)
call settextcolor
ret
disp44 endp
;this is bomb for level 2
bomb2 proc
call randomize
mov eax,row1
mov ebx,col1
mov tempi,eax
mov tempi1,ebx

mov eax,row1
mov ebx,rowlen2
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
cmp eax,0
je bombwipeforcolms
jne done
bombwipeforcolms:
inc reminder
mov col1,0
mov ecx,10
L1:
mov eax,row1
mov ebx,rowlen2
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
cmp eax,9
je equal5
jne equal6
equal5:
inc col1
jmp l1
```

```
equal6:
mov eax,5
call randomrange
cmp eax,0
je zero
jne notzero
zero:
inc eax
notzero:
mov level2[ebx+esi * indexsize2],eax
inc col1
dec cl
jnz l1

mov eax,tempi1
mov col1,eax
mov ecx,10
mov row1,0

again:
mov eax,row1
mov ebx,rowlen2
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
cmp eax,9
je equal7
jne equal8
equal7:
inc row1
jmp again
equal8:
mov eax,5
call randomrange
cmp eax,0
je zero1
jne notzero1
zero1:
inc eax
notzero1:
mov level2[ebx+esi * indexsize2],eax
inc row1
loop again
done:
mov eax,tempi
mov ebx,tempi1
mov row1,eax
mov col1,ebx
ret
bomb2 endp
;this autocombo function is for level2
autocombo2 proc

rowonecheck:
```

```
 mov row1,0
 mov col1,0
 call combolevel2
   mov row1,0
 mov col1,1
  call combolevel2
   mov row1,0
 mov col1,2
  call combolevel2
   mov row1,0
 mov col1,3
call combolevel2
   mov row1,0
 mov col1,4
  call combolevel2
   mov row1,0
 mov col1,5
call combolevel2
  mov row1,0
 mov col1,6
 call combolevel2
   mov row1,0
 mov col1,7
call combolevel2
   mov row1,0
 mov col1,8
 call combolevel2
   mov row1,0
 mov col1,9
call combolevel2


  rowtwocheck:
 mov row1,1
 mov col1,0
call combolevel2
   mov row1,1
 mov col1,1
call combolevel2
   mov row1,1
 mov col1,2
call combolevel2
   mov row1,1
 mov col1,3
call combolevel2
   mov row1,1
 mov col1,4
call combolevel2
   mov row1,1
 mov col1,5
call combolevel2
  mov row1,1
 mov col1,6
call combolevel2
```

```
   mov row1,1
 mov col1,7
call combolevel2
   mov row1,1
 mov col1,8
call combolevel2
   mov row1,1
 mov col1,9
call combolevel2
  rowthreecheck:
 mov row1,2
 mov col1,0
call combolevel2
   mov row1,2
 mov col1,1
call combolevel2
   mov row1,2
 mov col1,2
call combolevel2
   mov row1,2
 mov col1,3
call combolevel2
   mov row1,2
 mov col1,4
call combolevel2
   mov row1,2
 mov col1,5
call combolevel2
  mov row1,2
 mov col1,6
call combolevel2
   mov row1,2
 mov col1,7
call combolevel2
   mov row1,2
 mov col1,8
call combolevel2
   mov row1,2
 mov col1,9
call combolevel2
  rowfourcheck:
 mov row1,3
 mov col1,0
call combolevel2
   mov row1,3
 mov col1,1
call combolevel2
   mov row1,3
 mov col1,2
call combolevel2
   mov row1,3
 mov col1,3
call combolevel2
   mov row1,3
 mov col1,4
```

```
call combolevel2
   mov row1,3
 mov col1,5
call combolevel2
  mov row1,3
 mov col1,6
call combolevel2
   mov row1,3
 mov col1,7
call combolevel2
   mov row1,3
 mov col1,8
call combolevel2
   mov row1,3
 mov col1,9
call combolevel2
  rowfivecheck:
 mov row1,4
 mov col1,0
call combolevel2
   mov row1,4
 mov col1,1
call combolevel2
   mov row1,4
 mov col1,2
call combolevel2
   mov row1,4
 mov col1,3
call combolevel2
   mov row1,4
 mov col1,4
call combolevel2
   mov row1,4
 mov col1,5
call combolevel2
  mov row1,4
 mov col1,6
call combolevel2
   mov row1,4
 mov col1,7
call combolevel2
   mov row1,4
 mov col1,8
call combolevel2
   mov row1,4
 mov col1,9
call combolevel2
  rowsixcheck:
 mov row1,5
call combolevel2
   mov row1,5
 mov col1,1
call combolevel2
   mov row1,5
 mov col1,2
```

```
call combolevel2
   mov row1,5
 mov col1,3
call combolevel2
   mov row1,5
 mov col1,4
call combolevel2
   mov row1,5
 mov col1,5
call combolevel2
  mov row1,5
 mov col1,6
call combolevel2
   mov row1,5
 mov col1,7
call combolevel2
   mov row1,5
 mov col1,8
call combolevel2
   mov row1,5
 mov col1,9
call combolevel2
  rowseventhcheck:
 mov row1,6
 mov col1,0
call combolevel2
   mov row1,6
 mov col1,1
call combolevel2
   mov row1,6
 mov col1,2
call combolevel2
   mov row1,6
 mov col1,3
call combolevel2
   mov row1,6
 mov col1,4
call combolevel2
   mov row1,6
 mov col1,5
call combolevel2
  mov row1,6
 mov col1,6
call combolevel2
   mov row1,6
 mov col1,7
call combolevel2
   mov row1,6
 mov col1,8
call combolevel2
   mov row1,6
 mov col1,9
call combolevel2
  roweightcheck:
 mov row1,7
```

```
 mov col1,0
call combolevel2
   mov row1,7
 mov col1,1
call combolevel2
   mov row1,7
 mov col1,2
call combolevel2
   mov row1,7
 mov col1,3
call combolevel2
   mov row1,7
 mov col1,4
call combolevel2
   mov row1,7
 mov col1,5
call combolevel2
  mov row1,7
 mov col1,6
call combolevel2
   mov row1,7
 mov col1,7
call combolevel2
   mov row1,7
 mov col1,8
call combolevel2
   mov row1,7
 mov col1,9
call combolevel2
  rowninethcheck:
 mov row1,8
 mov col1,0
call combolevel2
   mov row1,8
 mov col1,1
call combolevel2
   mov row1,8
 mov col1,2
call combolevel2
   mov row1,8
 mov col1,3
call combolevel2
   mov row1,8
 mov col1,4
call combolevel2
   mov row1,8
 mov col1,5
call combolevel2
  mov row1,8
 mov col1,6
call combolevel2
   mov row1,8
 mov col1,7
call combolevel2
   mov row1,8
```

```
  mov col1,8
 call combolevel2
    mov row1,8
  mov col1,9
 call combolevel2
   rowtenthcheck:
  mov row1,9
  mov col1,0
 call combolevel2
    mov row1,9
  mov col1,1
   call combolevel2
    mov row1,9
  mov col1,2
 call combolevel2
    mov row1,9
  mov col1,3
 call combolevel2
    mov row1,9
  mov col1,4
 call combolevel2
    mov row1,9
  mov col1,5
 call combolevel2
   mov row1,9
  mov col1,6
 call combolevel2
    mov row1,9
  mov col1,7
 call combolevel2
    mov row1,9
  mov col1,8
 call combolevel2
    mov row1,9
  mov col1,9
 call combolevel2
  ret
autocombo2 endp
;this is for userdata
booluserdata1level2 proc
call level2userrow1
call level2usercol1
ret
booluserdata1level2 endp
;this is for user data
booluserdata2level2 proc

call level2userrow2
call level2usercol2
ret
booluserdata2level2 endp
;this check is for row if row is not adjecent
level2checkrow2 proc
next1:
mov eax,row1
```

```
mov ebx,row2
cmp eax,ebx ;if entered row1 and row2 are same!!
je l1
jne l2
l2:
mov eax,row1
mov ebx,row2
mov ecx,col1
mov edx,col2
mov temp,eax
mov temp1,ebx
mov temp2,ecx
mov temp3,edx

inc temp;next row
cmp temp,ebx
je l1q
jne l2q
l1q:
mov ecx,col1
mov edx,col2
cmp ecx,edx
je l11q
jne l22q
l11q:
jmp doneq
l22q:
jmp againinput
l2q:
mov eax,row1
mov ebx,row2
mov ecx,col1
mov edx,col2
mov temp,eax
mov temp1,ebx
mov temp2,ecx
mov temp3,edx
dec temp
cmp temp,ebx
je c1q
jne againinput
c1q:
mov ecx,col1
mov edx,col2
cmp ecx,edx
je c22q
jne againinput
c22q:
jmp doneq
l1:
mov eax,col1
mov ebx,col2
mov temp,eax
mov temp1,ebx
inc temp
```

```
cmp temp,ebx
je l3
jne l4
l3:
jmp done
dec temp
jmp done
l4:
mov eax,col1
mov ebx,col2
mov temp,eax

mov temp1,ebx
dec temp
cmp temp,ebx
je l5
jmp againinput
l5:
jmp done
againinput:
call disp33
mov edx,offset disp1
call writestring
call readint
mov row2,eax
call checkagain2

mov edx,offset disp2
call writestring
call readint
mov col2,eax
l101q:
cmp col2,0
jb l1xq
jmp l2xq
l1xq:
jmp l9xq
l2xq:
cmp col2,9
ja l3xq
jmp l5xq
l3xq:
jmp l9xq
l9xq:
call disp44
mov edx,offset disp2
call writestring
call readint
mov col2,eax
jmp l101q

l5xq:
jmp next1

mov eax,row2
```

```
mov ebx,rowlen
mul ebx
mov ebx,eax
mov esi,col2
mov eax,level2[ebx+esi * indexsize2]
mov valclm,eax
mov valindex3,ebx
mov valindex4,esi
call writedec
call crlf
call exchange2
done:
doneq:
ret
level2checkrow2 endp
;this check is for to check if row1 & column1 entered is not -1,and row&column entered is not 10
level2checkagain1 proc
cmp row1,0
jb l1
jmp l2
l1:
call disp44
call level2userrow1
l2:
cmp row1,9
ja l3
jmp l5
l3:
call disp44
call level2userrow1
l5:

ret
level2checkagain1 endp
;this check is for to check if row2 & column2 entered is not -1,and row&column entered is not 10
level2checkagain2 proc
cmp row2,0
jb l1
jmp l2
l1:
call disp44
call level2userrow2
l2:
cmp row2,9
ja l3
jmp l5
l3:
call disp44
call level2userrow2
l5:

ret
level2checkagain2 endp
;user row1
level2userrow1 proc
```

```
mov eax,yellow(black*16)
call settextcolor

mov edx,offset disp1
call writestring
call readint
mov row1,eax
call level2checkagain1
ret
level2userrow1 endp
;get user defined col1
level2usercol1 proc
mov eax,yellow(black*16)
call settextcolor
mov edx,offset disp2
call writestring
call readint
mov col1,eax
l101:
cmp col1,0
jb l1
jmp l2
l1:
jmp l7
l2:
cmp col1,9
ja l3
jmp l5
l3:
jmp l7
l7:
call disp44
mov eax,red(black*16)
call settextcolor
mov edx,offset disp2
call writestring
call readint
mov col1,eax
jmp l101
l5:
```

```
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov valindex1,ebx
mov valindex2,esi
call writedec
call crlf
mov var5,eax

ret
level2usercol1 endp
;get user defined row2
level2userrow2 proc
mov eax,yellow(black*16)
call settextcolor
mov edx,offset disp1
call writestring
call readint
mov row2,eax
call level2checkagain2
ret
level2userrow2 endp
;get user defined col2
level2usercol2 proc
mov eax,yellow(black*16)
call settextcolor
mov edx,offset disp2
call writestring
call readint
mov col2,eax

l101:
cmp col2,0
jb l1
jmp l2
l1:
jmp l9
l2:
cmp col2,9
ja l3
jmp l5
l3:
jmp l9
l9:
call disp44
mov eax,red(black*16)
call settextcolor
mov edx,offset disp2
call writestring
call readint
mov col2,eax
jmp l101
```

```
l5:
call level2checkrow2
jmp do1

do1:
mov eax,row2
mov ebx,rowlen2
mul ebx
mov ebx,eax
mov esi,col2
mov eax,level2[ebx+esi * indexsize2]
mov valclm,eax
mov valindex3,ebx
mov valindex4,esi
call writedec
call crlf
call exchange2
ret
level2usercol2 endp
;unexchange for level2
unexchange2 proc
mov ebx,valindex1
mov esi,valindex2
mov ecx,var5
cmp var5,0
je bm
jne bm1
bm:
call bomb2
jmp done
bm1:
mov level2[ebx+esi *indexsize2],ecx

mov ebx,valindex3
mov esi,valindex4
mov ecx,valclm
cmp valclm,0
je bm2
jne bm3
bm2:
call bomb2
jmp done
bm3:
mov level2[ebx+esi *indexsize2],ecx
done:
ret
unexchange2 endp
;level2 exchange
exchange2 proc
mov ebx,valindex1
mov esi,valindex2
mov ecx,valclm
mov level2[ebx+esi *indexsize2],ecx

mov ebx,valindex3
```

```
mov esi,valindex4
mov ecx,var5
mov level2[ebx+esi *indexsize2],ecx
call crlf
call bomb2
call level2notswapped
call manualcombolevel2

;call combolevel2
done:
ret
exchange2 endp
;combo for level2
manualcombolevel2 proc
call randomize
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checkagain
mov edx,9
cmp col1,edx
je check2
jne moveon
moveon:
mov edx,8
cmp col1,edx
je check2
jne moveon1
moveon1:
check1:

;checking nextrows after exchange
mov ecx,col1
inc ecx
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check2
jne continue
continue:

mov tempindex3,esi
mov tempval1,eax
inc ecx
```

```
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check2
jne continue1
continue1:
mov tempindex4,esi
mov tempval2,eax
compare:
mov eax,tempval1
cmp tempval,eax
je compare2
jne check2
compare2:
cmp eax,tempval2
je execute1
jne check2
execute1:
mov eax,5
call randomrange
cmp eax,0
je zero
jne notzero
zero:
inc eax
notzero:
mov ebx,tempindex1
mov esi,tempindex2
mov level2[ebx+esi * indexsize2],eax
mov esi,tempindex3
mov eax,5
call randomrange
cmp eax,0
je zero1
jne notzero1
zero1:
inc eax
notzero1:
mov level2[ebx+esi * indexsize2],eax
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero2
jne notzero2
zero2:
inc eax
notzero2:
mov level2[ebx+esi * indexsize2],eax
inc score
```

```
jmp done

mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax

check2:
;checking prevroes after exchange
mov ecx,col1
dec ecx
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check3
jne continue2
continue2:
mov tempindex3,esi
mov tempval1,eax
dec ecx
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check3
jne continue3
continue3:
mov tempindex4,esi
mov tempval2,eax
compare1check2:
mov eax,tempval1
cmp tempval,eax
je compare2check2
jne check3
compare2check2:
cmp eax,tempval2
je execute2
jne check3
execute2:
mov eax,5
call randomrange
```

```
cmp eax,0
je zero3
jne notzero3
zero3:
inc eax
notzero3:
mov ebx,tempindex1
mov esi,tempindex2
mov level2[ebx+esi * indexsize2],eax
mov esi,tempindex3
mov eax,5
call randomrange
cmp eax,0
je zero4
jne notzero4
zero4:
inc eax
notzero4:
mov level2[ebx+esi * indexsize2],eax
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero5
jne notzero5
zero5:
inc eax
notzero5:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done

mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax


check3:


;checking prevroes after exchange
mov ecx,row1
inc ecx
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
```

```
mov edx,9
cmp edx,eax
je check4
jne continue4
continue4:
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax
inc ecx
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check4
jne continue5
continue5:
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check3:
mov eax,tempval1
cmp tempval,eax
je compare2check3
jne check4
compare2check3:
cmp eax,tempval2
je execute3
jne check4
execute3:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero6
jne notzero6
zero6:
inc eax
notzero6:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero7
jne notzero7
zero7:
inc eax
notzero7:
mov level2[ebx+esi * indexsize2],eax
```

```
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero8
jne notzero8
zero8:
inc eax
notzero8:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done
check4:
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
dec ecx
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check5
jne continue6
continue6:
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax
dec ecx
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check5
jne continue7
continue7:
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
```

```
compare1check4:
mov eax,tempval1
cmp tempval,eax
je compare2check4
jne check5
compare2check4:
cmp eax,tempval2
je execute4
jne check5
execute4:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero9
jne notzero9
zero9:
inc eax
notzero9:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero10
jne notzero10
zero10:
inc eax
notzero10:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero11
jne notzero11
zero11:
inc eax
notzero11:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done

check5:

mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
```

```
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov edx,col1
inc edx
mov esi,edx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check6
jne continue8
continue8:
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax

mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov ecx,col1
dec ecx
mov esi,ecx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check6
jne continue9
continue9:
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check5:
mov eax,tempval1
cmp tempval,eax
je compare2check5
jne check6
compare2check5:
cmp eax,tempval2
je execute5
jne check6
execute5:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero12
jne notzero12
zero12:
```

```
inc eax
notzero12:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero13
jne notzero13
zero13:
inc eax
notzero13:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero14
jne notzero14
zero14:
inc eax
notzero14:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done


check6:

mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
inc ecx
mov ecx,ecx
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je done
jne continue10
continue10:
```

```
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax

mov edx,row1
dec edx
mov eax,rowlen2
mov ebx,edx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je done
jne continue11
continue11:
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check6:
mov eax,tempval1
cmp tempval,eax
je compare2check6
jne unswap
compare2check6:
cmp eax,tempval2
je execute6
jne unswap
execute6:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero15
jne notzero15
zero15:
inc eax
notzero15:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero16
jne notzero16
zero16:
inc eax
notzero16:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
```

```
call randomrange
cmp eax,0
je zero17
jne notzero17
zero17:
inc eax
notzero17:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done

unswap:
call unexchange2

done:
ret

manualcombolevel2 endp
;this is the automated combo for level 2
combolevel2 proc
call randomize
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checkagain
mov edx,9
cmp col1,edx
je check2
jne moveon
moveon:
mov edx,8
cmp col1,edx
je check2
jne moveon1
moveon1:
check1:

;checking nextrows after exchange
mov ecx,col1
inc ecx
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check2
```

```
jne continue
continue:

mov tempindex3,esi
mov tempval1,eax
inc ecx
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check2
jne continue1
continue1:
mov tempindex4,esi
mov tempval2,eax
compare:
mov eax,tempval1
cmp tempval,eax
je compare2
jne check2
compare2:
cmp eax,tempval2
je execute1
jne check2
execute1:
mov eax,5
call randomrange
cmp eax,0
je zero
jne notzero
zero:
inc eax
notzero:
mov ebx,tempindex1
mov esi,tempindex2
mov level2[ebx+esi * indexsize2],eax
mov esi,tempindex3
mov eax,5
call randomrange
cmp eax,0
je zero1
jne notzero1
zero1:
inc eax
notzero1:
mov level2[ebx+esi * indexsize2],eax
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero2
```

```
jne notzero2
zero2:
inc eax
notzero2:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done

mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax

check2:
;checking prevroes after exchange
mov ecx,col1
dec ecx
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check3
jne continue2
continue2:
mov tempindex3,esi
mov tempval1,eax
dec ecx
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check3
jne continue3
continue3:
mov tempindex4,esi
mov tempval2,eax
compare1check2:
mov eax,tempval1
cmp tempval,eax
je compare2check2
jne check3
compare2check2:
```

```
cmp eax,tempval2
je execute2
jne check3
execute2:
mov eax,5
call randomrange
cmp eax,0
je zero3
jne notzero3
zero3:
inc eax
notzero3:
mov ebx,tempindex1
mov esi,tempindex2
mov level2[ebx+esi * indexsize2],eax
mov esi,tempindex3
mov eax,5
call randomrange
cmp eax,0
je zero4
jne notzero4
zero4:
inc eax
notzero4:
mov level2[ebx+esi * indexsize2],eax
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero5
jne notzero5
zero5:
inc eax
notzero5:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done

mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax


check3:


;checking prevroes after exchange
mov ecx,row1
inc ecx
```

```
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check4
jne continue4
continue4:
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax
inc ecx
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check4
jne continue5
continue5:
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check3:
mov eax,tempval1
cmp tempval,eax
je compare2check3
jne check4
compare2check3:
cmp eax,tempval2
je execute3
jne check4
execute3:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero6
jne notzero6
zero6:
inc eax
notzero6:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
```

```
je zero7
jne notzero7
zero7:
inc eax
notzero7:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero8
jne notzero8
zero8:
inc eax
notzero8:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done
check4:
mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
dec ecx
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check5
jne continue6
continue6:
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax
dec ecx
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
```

```
je check5
jne continue7
continue7:
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check4:
mov eax,tempval1
cmp tempval,eax
je compare2check4
jne check5
compare2check4:
cmp eax,tempval2
je execute4
jne check5
execute4:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero9
jne notzero9
zero9:
inc eax
notzero9:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero10
jne notzero10
zero10:
inc eax
notzero10:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero11
jne notzero11
zero11:
inc eax
notzero11:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done

check5:

mov eax,rowlen2
```

```
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov edx,col1
inc edx
mov esi,edx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check6
jne continue8
continue8:
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax

mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov ecx,col1
dec ecx
mov esi,ecx
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je check6
jne continue9
continue9:
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check5:
mov eax,tempval1
cmp tempval,eax
je compare2check5
jne check6
compare2check5:
cmp eax,tempval2
je execute5
jne check6
execute5:
mov ebx,tempindex1
mov esi,tempindex2
```

```
mov eax,5
call randomrange
cmp eax,0
je zero12
jne notzero12
zero12:
inc eax
notzero12:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero13
jne notzero13
zero13:
inc eax
notzero13:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero14
jne notzero14
zero14:
inc eax
notzero14:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done


check6:

mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
inc ecx
mov ecx,ecx
mov eax,rowlen2
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
```

```
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je done
jne continue10
continue10:
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax

mov edx,row1
dec edx
mov eax,rowlen2
mov ebx,edx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
mov edx,9
cmp edx,eax
je done
jne continue11
continue11:
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check6:
mov eax,tempval1
cmp tempval,eax
je compare2check6
jne unswap
compare2check6:
cmp eax,tempval2
je execute6
jne unswap
execute6:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero15
jne notzero15
zero15:
inc eax
notzero15:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero16
jne notzero16
zero16:
```

```
inc eax
notzero16:
mov level2[ebx+esi * indexsize2],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero17
jne notzero17
zero17:
inc eax
notzero17:
mov level2[ebx+esi * indexsize2],eax
inc score
jmp done

unswap:
;call unexchange

done:
ret
combolevel2 endp
;the value 9 which cannot be swapped
level2notswapped proc
mov eax,row2
mov ebx,rowlen2
mul ebx
mov ebx,eax
mov esi,col2
mov eax,level2[ebx+esi * indexsize2]
cmp eax,9
je equal
jne unequal
equal:
mov edx,offset dispnotswap
call writestring
call crlf
call unexchange2
jmp done
unequal:

mov eax,rowlen2
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,level2[ebx+esi * indexsize2]
cmp eax,9
je equal1
jne unequal1
equal1:
mov edx,offset dispnotswap
call writestring
call crlf
```

```
call unexchange2
jmp done
unequal1:
done:
ret
level2notswapped endp
;this function is for x
cannotbeswapped proc
mov eax,row2
mov ebx,rowlen
mul ebx
mov ebx,eax
mov esi,col2
mov eax,array[ebx+esi * indexsize]
cmp eax,8
je equal
jne unequal
equal:
mov edx,offset dispnotswap
call writestring
call unexchange
jmp done
unequal:

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
cmp eax,8
je equal1
jne unequal1
equal1:
mov edx,offset dispnotswap
call writestring
call unexchange
jmp done
unequal1:
done:
ret
cannotbeswapped endp
;this function is for zero(0)
bomb proc
call randomize
mov eax,row1
mov ebx,col1
mov tempi,eax
mov tempi1,ebx

mov eax,row1
mov ebx,rowlen
mul ebx
mov ebx,eax
mov esi,col1
```

```
mov eax,array[ebx+esi * indexsize]
cmp eax,0
je bombwipeforcolms
jne done
bombwipeforcolms:
inc reminder
mov col1,0
mov ecx,10
L1:
mov eax,row1
mov ebx,rowlen
mul ebx
mov ebx,eax
mov esi,col1
mov eax,5
call randomrange
cmp eax,0
je zero
jne notzero
zero:
inc eax
notzero:
mov array[ebx+esi * indexsize],eax
inc col1
dec cl
jnz l1

mov eax,tempi1
mov col1,eax
mov ecx,10
mov row1,0
again:
mov eax,row1
mov ebx,rowlen
mul ebx
mov ebx,eax
mov esi,col1
mov eax,5
call randomrange
cmp eax,0
je zero1
jne notzero1
zero1:
inc eax
notzero1:
mov array[ebx+esi * indexsize],eax
inc row1
loop again
done:
mov eax,tempi
mov ebx,tempi1
mov row1,eax
mov col1,ebx
ret
bomb endp
```

```
;filehandling
filehandling proc

mov eax,yellow(black*16)
call settextcolor
mov edx,offset highscore
call writestring
call crlf
invoke createfile,addr filename,GENERIC_WRITE+GENERIC_READ,0,0,OPEN_ALWAYS,0,0
mov filehandle,eax
;invoke setfilepointer,filehandle,0,0,file_end
invoke writefile,filehandle,addr ldisp1,sizeof ldisp1,addr lcountdisp1,0
invoke writefile,filehandle,addr ldisp2,sizeof ldisp2,addr lcountdisp2,0
INVOKE SETFILEPOINTER,FILEHANDLE,0,0,FILE_BEGIN
INVOKE REAdFILE,FILEHANDLE,ADDR lDISP3,8,ADDR noofbytes,NULL
INVOKE SETFILEPOINTER,FILEHANDLE,0,0,FILE_CURRENT
INVOKE READFILE,FILEHANDLE,ADDR lDISP4,1,ADDR noofbytes,NULL
INVOKE READFILE,FILEHANDLE,ADDR lDISP4,4,ADDR noofbytes,NULL
mov ecx, sizeof lDISP4
mov ebx, 0
l1:
;cmp lDISP4[ebx], score[ebx]

loop l1
INVOKE CLOSEHANDLE,FILEHANDLE
MOV EDX,OFFSET lDISP3
CALL WRITESTRING
CALL CRLF
MOV EDX,OFFSET lDISP4
CALL WRITESTRING
CALL CRLF

ret
filehandling endp
;checkvaltostring
tostring proc
mov eax, score
mov ebx, varwithval
cmp eax, ebx
jg greater
je done
jmp less
greater:
mov edx, offset valGreater
call writestring
invoke createfile,addr filename,GENERIC_WRITE+GENERIC_READ,0,0,OPEN_ALWAYS,0,0
mov filehandle,eax
INVOKE SETFILEPOINTER,FILEHANDLE,0,0,FILE_END
invoke writefile,filehandle,score,3,addr lcountdisp2,0
INVOKE CloseHandle, fileHandle
jmp done
less:
mov edx, offset valSmaller
call writestring
done:
```

```
ret
tostring endp
;manual combo when there is no combo it wll unexchange
manualcombo proc

call randomize
mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checkagain
mov edx,9
cmp col1,edx
je check2
jne moveon
moveon:
mov edx,8
cmp col1,edx
je check2
jne moveon1
moveon1:
check1:

;checking nextrows after exchange
mov ecx,col1
inc ecx
mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,array[ebx+esi * indexsize]
mov tempindex3,esi
mov tempval1,eax
inc ecx
mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,array[ebx+esi * indexsize]
mov tempindex4,esi
mov tempval2,eax
compare:
mov eax,tempval1
cmp tempval,eax
je compare2
jne check2
compare2:
```

```
cmp eax,tempval2
je execute1
jne check2
execute1:
mov eax,5
call randomrange
cmp eax,0
je zero
jne notzero
zero:
inc eax
notzero:
mov ebx,tempindex1
mov esi,tempindex2
mov array[ebx+esi * indexsize],eax
mov esi,tempindex3
mov eax,5
call randomrange
cmp eax,0
je zero1
jne notzero1
zero1:
inc eax
notzero1:
mov array[ebx+esi * indexsize],eax
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero2
jne notzero2
zero2:
inc eax
notzero2:
mov array[ebx+esi * indexsize],eax
inc score
jmp done

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax

check2:
;checking prevroes after exchange
mov ecx,col1
dec ecx
mov eax,rowlen
mov ebx,row1
mul ebx
```

```
mov ebx,eax
mov esi,ecx
mov eax,array[ebx+esi * indexsize]
mov tempindex3,esi
mov tempval1,eax
dec ecx
mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,array[ebx+esi * indexsize]
mov tempindex4,esi
mov tempval2,eax
compare1check2:
mov eax,tempval1
cmp tempval,eax
je compare2check2
jne check3
compare2check2:
cmp eax,tempval2
je execute2
jne check3
execute2:
mov eax,5
call randomrange
cmp eax,0
je zero3
jne notzero3
zero3:
inc eax
notzero3:
mov ebx,tempindex1
mov esi,tempindex2
mov array[ebx+esi * indexsize],eax
mov esi,tempindex3
mov eax,5
call randomrange
cmp eax,0
je zero4
jne notzero4
zero4:
inc eax
notzero4:
mov array[ebx+esi * indexsize],eax
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero5
jne notzero5
zero5:
inc eax
notzero5:
mov array[ebx+esi * indexsize],eax
```

```
inc score
jmp done

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax


check3:


;checking prevroes after exchange
mov ecx,row1
inc ecx
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax
inc ecx
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check3:
mov eax,tempval1
cmp tempval,eax
je compare2check3
jne check4
compare2check3:
cmp eax,tempval2
je execute3
jne check4
execute3:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero6
```

```
jne notzero6
zero6:
inc eax
notzero6:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero7
jne notzero7
zero7:
inc eax
notzero7:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero8
jne notzero8
zero8:
inc eax
notzero8:
mov array[ebx+esi * indexsize],eax
inc score
jmp done
check4:


mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
dec ecx
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax
dec ecx
mov eax,rowlen
```

```
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check4:
mov eax,tempval1
cmp tempval,eax
je compare2check4
jne check5
compare2check4:
cmp eax,tempval2
je execute4
jne check5
execute4:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero9
jne notzero9
zero9:
inc eax
notzero9:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero10
jne notzero10
zero10:
inc eax
notzero10:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero11
jne notzero11
zero11:
inc eax
notzero11:
mov array[ebx+esi * indexsize],eax
inc score
jmp done

check5:
```

```
mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov edx,col1
inc edx
mov esi,edx
mov eax,array[ebx+esi * indexsize]
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov ecx,col1
dec ecx
mov esi,ecx
mov eax,array[ebx+esi * indexsize]
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check5:
mov eax,tempval1
cmp tempval,eax
je compare2check5
jne check6
compare2check5:
cmp eax,tempval2
je execute5
jne check6
execute5:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero12
jne notzero12
zero12:
inc eax
notzero12:
```

```
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero13
jne notzero13
zero13:
inc eax
notzero13:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero14
jne notzero14
zero14:
inc eax
notzero14:
mov array[ebx+esi * indexsize],eax
inc score
jmp done


check6:

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
inc ecx
mov ecx,ecx
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax

mov edx,row1
dec edx
mov eax,rowlen
```

```
mov ebx,edx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check6:
mov eax,tempval1
cmp tempval,eax
je compare2check6
jne unswap
compare2check6:
cmp eax,tempval2
je execute6
jne unswap
execute6:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero15
jne notzero15
zero15:
inc eax
notzero15:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero16
jne notzero16
zero16:
inc eax
notzero16:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero17
jne notzero17
zero17:
inc eax
notzero17:
mov array[ebx+esi * indexsize],eax
inc score
jmp done

unswap:
```

```
call unexchange

done:
ret
manualcombo endp
;this functions checks combo vertically and horizantallt
combo proc

call randomize
mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checkagain
mov edx,9
cmp col1,edx
je check2
jne moveon
moveon:
mov edx,8
cmp col1,edx
je check2
jne moveon1
moveon1:
check1:

;checking nextrows after exchange
mov ecx,col1
inc ecx
mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,array[ebx+esi * indexsize]
mov tempindex3,esi
mov tempval1,eax
inc ecx
mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,array[ebx+esi * indexsize]
mov tempindex4,esi
mov tempval2,eax
compare:
mov eax,tempval1
cmp tempval,eax
je compare2
```

```
jne check2
compare2:
cmp eax,tempval2
je execute1
jne check2
execute1:
mov eax,5
call randomrange
cmp eax,0
je zero
jne notzero
zero:
inc eax
notzero:
mov ebx,tempindex1
mov esi,tempindex2
mov array[ebx+esi * indexsize],eax
mov esi,tempindex3
mov eax,5
call randomrange
cmp eax,0
je zero1
jne notzero1
zero1:
inc eax
notzero1:
mov array[ebx+esi * indexsize],eax
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero2
jne notzero2
zero2:
inc eax
notzero2:
mov array[ebx+esi * indexsize],eax
inc score
jmp done

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax

check2:
;checking prevroes after exchange
mov ecx,col1
dec ecx
mov eax,rowlen
```

```
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,array[ebx+esi * indexsize]
mov tempindex3,esi
mov tempval1,eax
dec ecx
mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,ecx
mov eax,array[ebx+esi * indexsize]
mov tempindex4,esi
mov tempval2,eax
compare1check2:
mov eax,tempval1
cmp tempval,eax
je compare2check2
jne check3
compare2check2:
cmp eax,tempval2
je execute2
jne check3
execute2:
mov eax,5
call randomrange
cmp eax,0
je zero3
jne notzero3
zero3:
inc eax
notzero3:
mov ebx,tempindex1
mov esi,tempindex2
mov array[ebx+esi * indexsize],eax
mov esi,tempindex3
mov eax,5
call randomrange
cmp eax,0
je zero4
jne notzero4
zero4:
inc eax
notzero4:
mov array[ebx+esi * indexsize],eax
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero5
jne notzero5
zero5:
inc eax
```

```
notzero5:
mov array[ebx+esi * indexsize],eax
inc score
jmp done

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax


check3:


;checking prevroes after exchange
mov ecx,row1
inc ecx
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax
inc ecx
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check3:
mov eax,tempval1
cmp tempval,eax
je compare2check3
jne check4
compare2check3:
cmp eax,tempval2
je execute3
jne check4
execute3:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
```

```
cmp eax,0
je zero6
jne notzero6
zero6:
inc eax
notzero6:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero7
jne notzero7
zero7:
inc eax
notzero7:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero8
jne notzero8
zero8:
inc eax
notzero8:
mov array[ebx+esi * indexsize],eax
inc score
jmp done
check4:


mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
dec ecx
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax
```

```
dec ecx
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check4:
mov eax,tempval1
cmp tempval,eax
je compare2check4
jne check5
compare2check4:
cmp eax,tempval2
je execute4
jne check5
execute4:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero9
jne notzero9
zero9:
inc eax
notzero9:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero10
jne notzero10
zero10:
inc eax
notzero10:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero11
jne notzero11
zero11:
inc eax
notzero11:
mov array[ebx+esi * indexsize],eax
inc score
jmp done
```

```
check5:

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov edx,col1
inc edx
mov esi,edx
mov eax,array[ebx+esi * indexsize]
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov ecx,col1
dec ecx
mov esi,ecx
mov eax,array[ebx+esi * indexsize]
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check5:
mov eax,tempval1
cmp tempval,eax
je compare2check5
jne check6
compare2check5:
cmp eax,tempval2
je execute5
jne check6
execute5:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero12
jne notzero12
zero12:
```

```
inc eax
notzero12:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero13
jne notzero13
zero13:
inc eax
notzero13:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero14
jne notzero14
zero14:
inc eax
notzero14:
mov array[ebx+esi * indexsize],eax
inc score
jmp done


check6:

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex1,ebx
mov tempindex2,esi
mov tempval,eax
;checking prevroes after exchange
mov ecx,row1
inc ecx
mov ecx,ecx
mov eax,rowlen
mov ebx,ecx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex3,ebx
mov tempindex4,esi
mov tempval1,eax

mov edx,row1
```

```
dec edx
mov eax,rowlen
mov ebx,edx
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov tempindex5,ebx
mov tempindex6,esi
mov tempval2,eax
compare1check6:
mov eax,tempval1
cmp tempval,eax
je compare2check6
jne unswap
compare2check6:
cmp eax,tempval2
je execute6
jne unswap
execute6:
mov ebx,tempindex1
mov esi,tempindex2
mov eax,5
call randomrange
cmp eax,0
je zero15
jne notzero15
zero15:
inc eax
notzero15:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex3
mov esi,tempindex4
mov eax,5
call randomrange
cmp eax,0
je zero16
jne notzero16
zero16:
inc eax
notzero16:
mov array[ebx+esi * indexsize],eax
mov ebx,tempindex5
mov esi,tempindex6
mov eax,5
call randomrange
cmp eax,0
je zero17
jne notzero17
zero17:
inc eax
notzero17:
mov array[ebx+esi * indexsize],eax
inc score
jmp done
```

```
unswap:


done:
ret
combo endp
;this checks if they are combo in the board after its initalizing
autocombo proc
rowonecheck:
  mov row1,0
  mov col1,0
  call combo
    mov row1,0
  mov col1,1
  call combo
    mov row1,0
  mov col1,2
   call combo
    mov row1,0
  mov col1,3
   call combo
    mov row1,0
  mov col1,4
   call combo
    mov row1,0
  mov col1,5
  call combo
   mov row1,0
  mov col1,6
  call combo
    mov row1,0
  mov col1,7
  call combo
    mov row1,0
  mov col1,8
   call combo
    mov row1,0
  mov col1,9
   call combo


   rowtwocheck:
  mov row1,1
  mov col1,0
  call combo
    mov row1,1
  mov col1,1
  call combo
    mov row1,1
  mov col1,2
   call combo
    mov row1,1
  mov col1,3
   call combo
```

```
  mov row1,1
mov col1,4
 call combo
  mov row1,1
mov col1,5
call combo
 mov row1,1
mov col1,6
call combo
  mov row1,1
mov col1,7
call combo
  mov row1,1
mov col1,8
 call combo
  mov row1,1
mov col1,9
 call combo
 rowthreecheck:
mov row1,2
mov col1,0
call combo
  mov row1,2
mov col1,1
call combo
  mov row1,2
mov col1,2
 call combo
  mov row1,2
mov col1,3
 call combo
  mov row1,2
mov col1,4
 call combo
  mov row1,2
mov col1,5
call combo
 mov row1,2
mov col1,6
call combo
  mov row1,2
mov col1,7
call combo
  mov row1,2
mov col1,8
 call combo
  mov row1,2
mov col1,9
 call combo
 rowfourcheck:
mov row1,3
mov col1,0
call combo
  mov row1,3
mov col1,1
```

```
call combo
  mov row1,3
mov col1,2
 call combo
  mov row1,3
mov col1,3
 call combo
  mov row1,3
mov col1,4
 call combo
  mov row1,3
mov col1,5
call combo
 mov row1,3
mov col1,6
call combo
  mov row1,3
mov col1,7
call combo
  mov row1,3
mov col1,8
 call combo
  mov row1,3
mov col1,9
 call combo
 rowfivecheck:
mov row1,4
mov col1,0
call combo
  mov row1,4
mov col1,1
call combo
  mov row1,4
mov col1,2
 call combo
  mov row1,4
mov col1,3
 call combo
  mov row1,4
mov col1,4
 call combo
  mov row1,4
mov col1,5
call combo
 mov row1,4
mov col1,6
call combo
  mov row1,4
mov col1,7
call combo
  mov row1,4
mov col1,8
 call combo
  mov row1,4
mov col1,9
```

```
 call combo
 rowsixcheck:
mov row1,5
mov col1,0
call combo
  mov row1,5
mov col1,1
call combo
  mov row1,5
mov col1,2
 call combo
  mov row1,5
mov col1,3
 call combo
  mov row1,5
mov col1,4
 call combo
  mov row1,5
mov col1,5
call combo
 mov row1,5
mov col1,6
call combo
  mov row1,5
mov col1,7
call combo
  mov row1,5
mov col1,8
 call combo
  mov row1,5
mov col1,9
 call combo
 rowseventhcheck:
mov row1,6
mov col1,0
call combo
  mov row1,6
mov col1,1
call combo
  mov row1,6
mov col1,2
 call combo
  mov row1,6
mov col1,3
 call combo
  mov row1,6
mov col1,4
 call combo
  mov row1,6
mov col1,5
call combo
 mov row1,6
mov col1,6
call combo
  mov row1,6
```

```
mov col1,7
call combo
  mov row1,6
mov col1,8
 call combo
  mov row1,6
mov col1,9
 call combo
 roweightcheck:
mov row1,7
mov col1,0
call combo
  mov row1,7
mov col1,1
call combo
  mov row1,7
mov col1,2
 call combo
  mov row1,7
mov col1,3
 call combo
  mov row1,7
mov col1,4
 call combo
  mov row1,7
mov col1,5
call combo
 mov row1,7
mov col1,6
call combo
  mov row1,7
mov col1,7
call combo
  mov row1,7
mov col1,8
 call combo
  mov row1,7
mov col1,9
 call combo
 rowninethcheck:
mov row1,8
mov col1,0
call combo
  mov row1,8
mov col1,1
call combo
  mov row1,8
mov col1,2
 call combo
  mov row1,8
mov col1,3
 call combo
  mov row1,8
mov col1,4
 call combo
```

```
  mov row1,8
 mov col1,5
 call combo
  mov row1,8
 mov col1,6
 call combo
  mov row1,8
 mov col1,7
 call combo
  mov row1,8
 mov col1,8
  call combo
  mov row1,8
 mov col1,9
  call combo
  rowtenthcheck:
 mov row1,9
 mov col1,0
 call combo
  mov row1,9
 mov col1,1
 call combo
  mov row1,9
 mov col1,2
  call combo
  mov row1,9
 mov col1,3
  call combo
  mov row1,9
 mov col1,4
  call combo
  mov row1,9
 mov col1,5
 call combo
  mov row1,9
 mov col1,6
 call combo
  mov row1,9
 mov col1,7
 call combo
  mov row1,9
 mov col1,8
  call combo
  mov row1,9
 mov col1,9
  call combo
 ret
autocombo endp
;check for if the next entered index is adjecent or not
checkrow2 proc
next1:
mov eax,row1
mov ebx,row2
cmp eax,ebx ;if entered row1 and row2 are same!!
je l1
```

```
jne l2
l2:
mov eax,row1
mov ebx,row2
mov ecx,col1
mov edx,col2
mov temp,eax
mov temp1,ebx
mov temp2,ecx
mov temp3,edx

inc temp;next row
cmp temp,ebx
je l1q
jne l2q
l1q:
mov ecx,col1
mov edx,col2
cmp ecx,edx
je l11q
jne l22q
l11q:
jmp doneq
l22q:
jmp againinput
l2q:
mov eax,row1
mov ebx,row2
mov ecx,col1
mov edx,col2
mov temp,eax
mov temp1,ebx
mov temp2,ecx
mov temp3,edx
dec temp
cmp temp,ebx
je c1q
jne againinput
c1q:
mov ecx,col1
mov edx,col2
cmp ecx,edx
je c22q
jne againinput
c22q:
jmp doneq
l1:
mov eax,col1
mov ebx,col2
mov temp,eax
mov temp1,ebx
inc temp
cmp temp,ebx
je l3
jne l4
```

```
l3:
jmp done
dec temp
jmp done
l4:
mov eax,col1
mov ebx,col2
mov temp,eax
mov temp1,ebx
dec temp
cmp temp,ebx
je l5
jmp againinput
l5:
jmp done
againinput:
call disp33

mov edx,offset disp1
call writestring
call readint
mov row2,eax
call checkagain2

mov edx,offset disp2
call writestring
call readint
mov col2,eax
l101q:
cmp col2,0
jb l1xq
jmp l2xq
l1xq:
jmp l9xq
l2xq:
cmp col2,9
ja l3xq
jmp l5xq
l3xq:
jmp l9xq
l9xq:
call disp44
mov edx,offset disp2
call writestring
call readint
mov col2,eax
jmp l101q

l5xq:
jmp next1

mov eax,row2
mov ebx,rowlen
mul ebx
mov ebx,eax
```

```
mov esi,col2
mov eax,array[ebx+esi * indexsize]
mov valclm,eax
mov valindex3,ebx
mov valindex4,esi
call writedec
call crlf
call exchange1
done:
doneq:
ret
checkrow2 endp
;this displays the score
scoredisp proc

mov eax,yellow(black*16)
call settextcolor
mov edx,offset dispnam
call writestring
mov edx,offset namee
call writestring
mov eax,green(black*16)
call settextcolor
mov edx,offset scoredisp1
call writestring
mov eax,score
call writedec
mov eax,red(black*16)
call settextcolor
mov edx,offset spa
call writestring
mov eax,moves
call writedec
mov eax,white(black*16)
call settextcolor
ret
scoredisp endp
;this check is for to check if row1 & column1 entered is not -1,and row&column entered is not 10
checkagain1 proc
cmp row1,0
jb l1
jmp l2
l1:
call disp44
call userrow1
l2:
cmp row1,9
ja l3
jmp l5
l3:
call disp44
call userrow1
l5:

ret
```

```
checkagain1 endp
;this check is for to check if row2 & column2 entered is not -1,and row&column entered is not 10
checkagain2 proc
cmp row2,0
jb l1
jmp l2
l1:
call disp44
call userrow2
l2:
cmp row2,9
ja l3
jmp l5
l3:
call disp44
call userrow2
l5:

ret
checkagain2 endp
;intializing the 10x10 matrix with randomvalues
initial proc
call randomize
 call crlf
    mov esi,offset array
    MOV ecx, 100
L1:

    mov eax,6 ; this is for random range
   call randomrange
  cmp eax,0
  je c1
  jne c2
  c1:
  inc eax
  c2:
 mov [esi],eax
 add esi,indexsize
DEC CL
JNZ L1

mov esi,offset array
    MOV ecx, 10
L2:

    mov eax,6 ; this is for random range
   call randomrange
   mov row2,eax
   mov col2,eax
   mov eax,row2
mov ebx,rowlen
mul ebx
mov ebx,eax
mov esi,col2
mov array[ebx+esi * indexsize],0
```

```
DEC CL
JNZ L2

    ret
initial endp
;procedure for displaying the matrix of level1
display proc
call scoredisp
call crlf
call crlf
mov eax,lightblue(black*16)
call settextcolor
mov ebx,offset array
mov esi,offset array
mov ecx,10
mov esi,0
mov edx,offset str1
 l1:
 push ecx
 mov ecx,10
 mov esi,0
 l2:
 mov eax,0
 mov edx,offset str1
 call writestring
 mov eaX,[ebx+esi]
 call writedec
 mov edx,offset str2
 call writestring
 mov edx,offset str1
 call writestring
 add esi,indexsize
 loop l2
 call crlf
 mov edx,offset space
 call writestring
 call crlf
 call crlf
 add ebx,rowlen
 pop ecx
 loop l1
ret
display endp
;this is to input player name
inputname proc
mov eax,yellow(black*16)
call settextcolor
mov edx,offset stt1
call writestring
mov edx,offset namee
mov ecx,sizeof namee
call readstring
mov nameecount,eax
call writestring
call crlf
```

```
ret
inputname endp
;get user defined row1
userrow1 proc
mov eax,yellow(black*16)
call settextcolor

mov edx,offset disp1
call writestring
call readint
mov row1,eax
call checkagain1
ret
userrow1 endp
;get user defined col1
usercol1 proc
mov eax,yellow(black*16)
call settextcolor
mov edx,offset disp2
call writestring
call readint
mov col1,eax
l101:
cmp col1,0
jb l1
jmp l2
l1:
jmp l7
l2:
cmp col1,9
ja l3
jmp l5
l3:
jmp l7
l7:
call disp44
mov edx,offset disp2
call writestring
call readint
mov col1,eax
jmp l101
l5:

mov eax,rowlen
mov ebx,row1
mul ebx
mov ebx,eax
mov esi,col1
mov eax,array[ebx+esi * indexsize]
mov valindex1,ebx
mov valindex2,esi
call writedec
call crlf
mov var5,eax
```

```
ret
usercol1 endp
;get user defined row2
userrow2 proc
mov eax,yellow(black*16)
call settextcolor
mov edx,offset disp1
call writestring
call readint
mov row2,eax
call checkagain2
ret
userrow2 endp
;get user defined col2
usercol2 proc
mov eax,yellow(black*16)
call settextcolor
mov edx,offset disp2
call writestring
call readint
mov col2,eax

l101:
cmp col2,0
jb l1
jmp l2
l1:
jmp l9
l2:
cmp col2,9
ja l3
jmp l5

l3:
jmp l9
l9:
call disp44
mov edx,offset disp2
call writestring
call readint
mov col2,eax
jmp l101
l5:
call checkrow2
jmp do1

do1:
mov eax,row2
mov ebx,rowlen
mul ebx
mov ebx,eax
mov esi,col2
mov eax,array[ebx+esi * indexsize]
mov valclm,eax
mov valindex3,ebx
```

```
mov valindex4,esi
call writedec
call crlf
call exchange1
ret
usercol2 endp
;function of bool to get input of row1 and col 1
booluser1data proc
call userrow1
call usercol1
ret
booluser1data endp
;funtion of bool to get input of row2 and col2
booluser2data proc
call userrow2
call usercol2
ret
booluser2data endp
;function to swap the values
exchange1 proc
mov ebx,valindex1
mov esi,valindex2
mov ecx,valclm
mov array[ebx+esi *indexsize],ecx

mov ebx,valindex3
mov esi,valindex4
mov ecx,var5
mov array[ebx+esi *indexsize],ecx
call crlf
call cannotbeswapped
call bomb
call manualcombo
done:
ret
exchange1 endp
;function to unswap the values
unexchange proc
mov ebx,valindex1
mov esi,valindex2
mov ecx,var5
cmp var5,0
je bm
jne bn
bm:
call bomb
jmp done
bn:
mov array[ebx+esi *indexsize],ecx

mov ebx,valindex3
mov esi,valindex4
mov ecx,valclm
cmp valclm,0
je bm1
```

```
jne bn1
bm1:
call bomb
jmp done
bn1:
mov array[ebx+esi *indexsize],ecx
done:
ret
unexchange endp
;random values initializer for level2
initial2 proc
call randomize
 call crlf
    mov esi,offset level2
    MOV ecx, 100
L1:
  mov eax,6 ; this is for random range
   call randomrange
  cmp eax,0
  je c1
  jne c2
  c1:
  inc eax
  c2:
mov ebx,[esi]
cmp ebx,9
je equal
jne unequal
equal:
add esi,indexsize2
jmp done
unequal:
mov [esi],eax
add esi,indexsize2
done:
DEC CL
JNZ L1
L2:

   mov eax,6 ; this is for random range
   call randomrange
   mov row2,eax
  ; inc eax
   mov col2,eax
   mov eax,row2
mov ebx,rowlen
mul ebx
mov ebx,eax
mov esi,col2
mov eax,level2[ebx+esi * indexsize2]
cmp eax,9
je equal3
jne equal4
equal3:
add esi,indexsize2
```

```
mov level2[ebx+esi * indexsize2],0
jmp lok1
equal4:
mov level2[ebx+esi * indexsize2],0
lok1:
DEC CL
JNZ L2
ret
initial2 endp
;display of level2
display2 proc
call scoredisp
mov eax,magenta(black*16)
call settextcolor
call crlf
mov ebx,offset level2
mov esi,offset level2
mov ecx,10
mov esi,0
mov edx,offset str1
 l1:
 push ecx
 mov ecx,10
 mov esi,0
 l2:
 mov eax,0
 mov edx,offset str1
 call writestring
 mov eaX,[ebx+esi]
 call writedec
 mov edx,offset str2
 call writestring
 mov edx,offset str1
 call writestring
 add esi,indexsize2
 loop l2
 call crlf
 mov edx,offset space
 call writestring
 call crlf
 call crlf
 add ebx,rowlen2
 pop ecx
loop l1
ret
display2 endp
;display of level3
display3 proc

mov eax,yellow(black*16)
call settextcolor
call scoredisp
call crlf
call crlf
mov eax,gray(black*16)
```

```
call settextcolor
mov ebx,offset array
mov esi,offset array
mov ecx,10
mov esi,0
mov edx,offset str1
 l1:
 push ecx
 mov ecx,10
 mov esi,0
 l2:
 mov eax,0
 mov edx,offset str1
 call writestring
 mov eaX,[ebx+esi]
 call writedec
 mov edx,offset str2
 call writestring
 mov edx,offset str1
 call writestring
 add esi,indexsize
 loop l2
 call crlf
 mov edx,offset space
 call writestring
 call crlf
 add ebx,rowlen
 pop ecx
 loop l1
ret
display3 endp
;random values initializer for level2
initial3 proc
call randomize
 call crlf
    mov esi,offset array
    MOV ecx, 100
L1:

    mov eax,6 ; this is for random range
   call randomrange
  cmp eax,0
  je c1
  jne c2
  c1:
  inc eax
  c2:
 mov [esi],eax
 add esi,indexsize
DEC CL
JNZ L1

mov esi,offset array
    MOV ecx, 10
L2:
```

```
    mov eax,6 ; this is for random range
   call randomrange
   mov row2,eax
   inc eax
   mov col2,eax
   mov eax,row2
mov ebx,rowlen
mul ebx
mov ebx,eax
mov esi,col2
mov array[ebx+esi * indexsize],0
DEC CL
JNZ L2
L3:

    mov eax,6 ; this is for random range
   call randomrange
   mov row2,eax
   mov col2,eax
   mov eax,row2
mov ebx,rowlen
mul ebx
mov ebx,eax
mov esi,col2
mov eax,8
mov array[ebx+esi * indexsize],eax
DEC CL
JNZ L3
    ret
initial3 endp
;start menu
start proc
mov eax,cyan(black*16)
call settextcolor
mov edx,offset dp
call writestring
call crlf
mov edx,offset dp1
call writestring
call crlf
mov edx,offset dp2
call writestring
call crlf
call readdec
ret
start endp
;while ending to check if you mistakenly entered to exit
areusure proc
mov eax,cyan(black*16)
call settextcolor

mov edx,offset dp3
call writestring
call crlf
```

```
mov edx,offset dp4
call writestring
call crlf
mov edx,offset dp5
call writestring
call crlf
call readdec
ret
areusure endp
;this functions is helper function of main functions!!
run proc
call filehandling
call crlf
call crlf
call start
mov ebx,eax
cmp eax,1
je proceed

jne notproced
proceed:
;starting of level1
call inputname
call initial
call autocombo
call autocombo
call clrscr
l99:
call autocombo
call display
call booluser1data
call booluser2data
inc moves
call autocombo
call clrscr
mov eax,15
cmp eax,moves
je newlevel
jne samelevel
samelevel:
call l99
newlevel:
mov moves,0
;starting of level2
call initial2
call autocombo2
call clrscr
l808:
call display2
call booluserdata1level2
call booluserdata2level2
inc moves
call autocombo2
call clrscr
mov eax,15
```

```
cmp eax,moves
je nextlevel
jne pro1
pro1:
call l808
nextlevel:
mov moves,0
call initial3
call autocombo
call clrscr
l980:
call autocombo
call display3
call booluser1data
call booluser2data
inc moves
call autocombo
call clrscr

call l980
;if you are sure to exit the game
notproced:
call areusure
mov ebx,eax
cmp ebx,2
je done
jne proceed
done:
mov eax,cyan(black*16)
call settextcolor
mov edx,offset dp6
call writestring
call crlf
;this function will check if our score is greater then the already exsisting score
call tostring
ret
run endp
;main needs no introduction!!
main proc
call run

 exit
main endp
end main
```