.ORIG x3000

LD R0, BEGIN;

LDR R1, R0 ,#0;

LDR R2, R0, #1;

ST R1, N; store the row of Matrix

ST R2, M; store the column of Matrix

NOT R4, R2;

ADD R4, R4, #1;

ST R4 ,Negative\_M;

NOT R4, R1;

ADD R4, R4, #1;

ST R4, Negative\_N;

AND R3, R3, #0;

ST R3 MAX; reset the MAX number

LOOP1 ADD R3, R3,R1;

ADD R2, R2, #-1;

BRp LOOP1; R3 is the N\*M

ADD R0, R0, R3;

ADD R0, R0, #1;

ST R0, LAST; store the last data's address

NOT R0, R0;

ADD R0, R0, #1;

ST R0, Negative\_LAST; store the negative last address

LD R6, STACK; now R6 is the ptr of stack

AND R0, R0, #0; clear the value of output address

LEA R1,ANSWER;

LOOP2 STR R0, R1, #0;

ADD R1,R1, #1;

ADD R3, R3, #-1;

BRp LOOP2;

;init all the register

AND R3, R3, #0;

AND R1, R1, #0;

AND R2, R2, #0;

AND R0, R0, #0;

AND R4, R4, #0;

AND R5, R5, #0;

LD R1, FIRST;

Each\_Begin LD R4, LAST;

NOT R4, R4;

ADD R4, R4, #1;

ADD R4, R1, R4;

BRp BREAK; every address is checked

AND R4, R4, #0;

ADD R4, R4, #1;

AND R3, R3, #0;

AND R5, R5, #0;

AND R2, R2, #0;

JSR DFS; run dfs and storage the output

ADD R1, R1 ,#1;

BRnzp Each\_Begin;

DFS LD R2, MAX; if R4 > MAX then MAX=R4

NOT R2, R2;

ADD R2,R2, #1;

ADD R2, R2, R4; check which is bigger;

BRnz #1;

ST R4, MAX;

RIGHT LD R2,FIRST;

NOT R2,R2;

ADD R2, R2, #2;

ADD R2, R2, R1;

LD R3, Negative\_M;

RIGHT\_LOOP ADD R2, R2,R3; whether can divide exactly

BRz LEFT; check left

BRp RIGHT\_LOOP;

;now we check whether can go to the right;

LDR R3, R1,#0;

NOT R3, R3;

ADD R3, R3, #1; -Value of R1

ADD R2, R1, #1;

LDR R2, R2, #0;

ADD R2, R2, R3; check which is higher

BRnz LEFT;

ADD R2, R1, #1;

LEA R5, LEFT;

;put now station into stack

ADD R6, R6, #-1;

STR R1, R6, #0;

ADD R6, R6, #-1;

STR R4, R6, #0;

ADD R6, R6, #-1;

STR R5, R6, #0;

ADD R1, R2, #0;

ADD R4, R4, #1;

BRnzp DFS;

LEFT LD R3, Negative\_M;

LD R2,FIRST;

NOT R2,R2;

ADD R2, R2, #1;

ADD R2, R2, R1;

LEFT\_LOOP BRz TOP;

ADD R2, R2, R3;

BRzp LEFT\_LOOP;

;now we check whether can go to the right;

LDR R3, R1,#0;

NOT R3, R3;

ADD R3, R3, #1; -Value of R1

ADD R2, R1, #-1;

LDR R2, R2, #0;

ADD R2, R2, R3; check which is higher

BRnz TOP;

ADD R2, R1, #-1;

LEA R5, TOP;

;put now station into stack

ADD R6, R6, #-1;

STR R1, R6, #0;

ADD R6, R6, #-1;

STR R4, R6, #0;

ADD R6, R6, #-1;

STR R5, R6, #0;

ADD R1, R2, #0;

ADD R4, R4, #1;

BRnzp DFS;

TOP LD R2,FIRST;

LD R3, M;

ADD R2, R2, R3;

NOT R2, R2;

ADD R2, R2, #1;

ADD R2, R2, R1;

BRn BOTTOM;

;now we check whether can go to the right;

LDR R3, R1,#0;

NOT R3, R3;

ADD R3, R3, #1; -Value of R1

LD R2, Negative\_M;

ADD R2, R1, R2;

LDR R2, R2, #0;

ADD R2, R2, R3; check which is higher

BRnz BOTTOM;

LD R2, Negative\_M;

ADD R2, R1, R2; R2 is the now address

LEA R5, BOTTOM;

;put now station into stack

ADD R6, R6, #-1;

STR R1, R6, #0;

ADD R6, R6, #-1;

STR R4, R6, #0;

ADD R6, R6, #-1;

STR R5, R6, #0;

ADD R1, R2, #0;

ADD R4, R4, #1;

BRnzp DFS;

BOTTOM LD R2,Negative\_LAST;

LD R3, M;

ADD R3,R3, R1;

ADD R2, R2, R3;

BRp RETURN;

;now we check whether can go to the right;

LDR R3, R1,#0;

NOT R3, R3;

ADD R3, R3, #1; -Value of R1

LD R2, M;

ADD R2, R1, R2;

LDR R2, R2, #0; R2 now is address

ADD R2, R2, R3; check which is higher

BRnz RETURN;

LD R2, M;

ADD R2, R1, R2; R2 is the now address

LEA R5, RETURN;

;put now station into stack

ADD R6, R6, #-1;

STR R1, R6, #0;

ADD R6, R6, #-1;

STR R4, R6, #0;

ADD R6, R6, #-1;

STR R5, R6, #0;

ADD R1, R2, #0;

ADD R4, R4, #1;

BRnzp DFS;

;now it's time to RETURN

RETURN LD R2, Negative\_STACK;

ADD R2, R2, R6;

BRnp #1; if R6 is x3000 then return ;

RET;

LDR R5, R6, #0;

ADD R6, R6, #1;

LDR R4, R6, #0;

ADD R6, R6, #1;

LDR R1, R6, #0;

ADD R6, R6, #1;

JMP R5;

;now DFS is over and output the needed number

BREAK LD R2, MAX;

HALT;

N .FILL #1;

M .FILL #1;

Negative\_N .FILL #1;

Negative\_M .FILL #1;

MAX .FILL #-1;

ANSWER .BLKW #50;

STACK .FILL x8000;

Negative\_STACK .FILL x-8000;

BEGIN .FILL x4000;

FIRST .FILL x4002;

LAST .FILL #1;

Negative\_LAST .FILL #1;

.END

.ORIG x4000

.FILL #7

.FILL #7

.FILL #100

.FILL #99

.FILL #98

.FILL #97

.FILL #96

.FILL #95

.FILL #94

.FILL #77

.FILL #76

.FILL #75

.FILL #74

.FILL #73

.FILL #72

.FILL #93

.FILL #78

.FILL #61

.FILL #60

.FILL #59

.FILL #58

.FILL #71

.FILL #92

.FILL #79

.FILL #62

.FILL #53

.FILL #52

.FILL #57

.FILL #70

.FILL #91

.FILL #80

.FILL #63

.FILL #54

.FILL #55

.FILL #56

.FILL #69

.FILL #90

.FILL #81

.FILL #64

.FILL #65

.FILL #66

.FILL #67

.FILL #68

.FILL #89

.FILL #82

.FILL #83

.FILL #84

.FILL #85

.FILL #86

.FILL #87

.FILL #88

.END